AFIT/GCS/ENG/95D-04

PROBABILISTIC KNOWLEDGE BASE VALIDATION

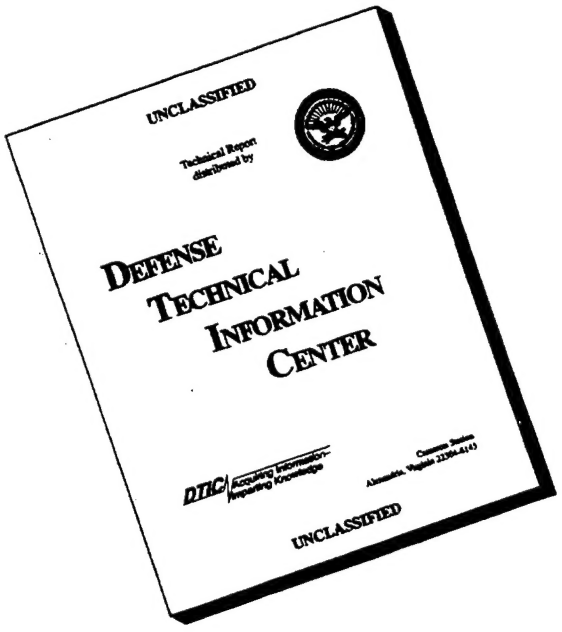
THESIS

Howard Terrance Gleason
Captain

AFIT/GCS/ENG/95D-04

Approved for public release; distribution unlimited

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GCS/ENG/95D-04

PROBABILISTIC KNOWLEDGE BASE VALIDATION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Howard Terrance Gleason, B.S.

Captain

December, 1995

Acknowledgements

This has been an extremely humbling experience. I truly admire anyone who makes a living performing and writing about research.

I want to thank my advisor Dr. Santos for all the guidance and motivation I had in producing this research.

I also want to thank the rest of my committee, Lt. Col. Bailor and Dr. Potoczny, for providing excellent suggestions on improving the document.

Thanks to Shawn for teaming up and taking on Doc with me.

Thanks to all my classmates for putting up with my endless barrage of questions and general verbosity during my time here.

A special thanks to Stan for providing many hours of challenging discussions, and working with me on a daily basis.

Thanks to all my family for supporting me and tolerating all the disruptions as I completed my degree.

Most of all I have to thank my gorgeous wife, Carolyn. Without all her support, without all her understanding, and without all her cooking, I would have never made it through this program. Thanks, Honey!

Howard Terrance Gleason

Table of Contents

List of Figures

List of Tables

Abstract

Our work develops a new methodology and tool for the validation of probabilistic knowledge bases throughout their lifecycle. The methodology minimizes user interaction by automatically modifying incorrect knowledge; only the occurrence of incomplete knowledge involves interaction. These gains are realized by combining and modifying techniques borrowed from rule-based and artificial neural network validation strategies. The presented methodology is demonstrated through BVAL, which is designed for a new knowledge representation, the Bayesian Knowledge Base. This knowledge representation accommodates incomplete knowledge while remaining firmly grounded in probability theory.

PROBABILISTIC KNOWLEDGE BASE VALIDATION

I. Introduction

The development of Knowledge-Based Systems (KBSs[1]) began in 1965 with DENDRAL [30], a complex computer system that aided research chemists in molecular structure elucidation[2]. From DENDRAL came some of the classic KBSs: MYCIN [6] (blood disease diagnosis), PROSPECTOR [13] (geologic mineral exploration), and XCON/R1 [31] (computer system configuration). Development and use of Knowledge-Based Systems has steadily increased since their inception, and seems likely to continue[16][17].

Any system, including a KBS, is of little use if it does not function properly. While many components of a KBS contribute to its overall "correctness" (user interface, inference engine, knowledge base), this research is concerned with the correctness of the knowledge base, not the entire knowledge-based system. Specifically, the critical factor focused on here is ensuring that the knowledge base provides an acceptable answer to every query. In the medical domain, this is analogous to ensuring the doctor provides the correct diagnosis, without regard for bedside manner, location of office, hourly charges, etc.. This process is generally termed the "verification and validation (V&V) of the knowledge base." Verification can be thought of as ensuring the knowledge is collected and structured properly, while validation ensures the knowledge produces correct results.

Historically, the development and application of knowledge-based systems has been published before any V&V results on the same system [36, 6], and the trend continues today. Furthermore,

[1] As the development and use of *expert* systems expanded, some systems made use of non-expert knowledge. The need to refer to systems utilizing a separate knowledge and algorithm paradigm, but not necessarily making use of "expert knowledge", resulted in the more general term: Knowledge-Based Systems (KBSs). Although the distinction between expert systems and KBSs is meaningless to some authors, our work uses the term KBS to emphasize the general applicability of its contents.

[2] Although many authors credit MYCIN as the first KBS since it clearly separated the knowledge from the algorithms, as [30] points out, DENDRAL was the first to exhibit those characteristics, if only in part.

while KBSs using probabilistic knowledge representations are popular, little literature exists regarding the validation or verification of those knowledge bases. It is this missing piece of information that our work addresses.

Chapter II describes current approaches to V&V from the rule-based and neural network communities. These approaches show that current techniques rely heavily on the special traits of their individual knowledge representations. Building on this, Chapter III describes the Bayesian Knowledge Base (BKB). This probabilistic knowledge representation is the structure that gives rise to BVAL.[3] Chapter IV discusses the new validation methodology developed for BKBs and the tool (BVAL) developed to apply this methodology. Conclusions derived from this research are presented in Chapter V, with possibilities for future research discussed in Chapter VI.

[3]The Bayesian Knowledge Base and BVAL were developed for a new KBS architecture called PESKI. See Appendix A.

II. V&V Background

Ensuring acceptable performance of any system is necessary for the success of that system. Unique to knowledge-based systems, is the separation between the knowledge contained in the system and the algorithms used to reason with and apply that knowledge. Thus KBSs can be evaluated on at least two different levels: the knowledge level and the system level. A system level evaluation not only takes into account the knowledge in the system, but also the inferencing mechanism, the knowledge acquisition mechanism, and the user-interface. This overall, system-wide, evaluation is necessary for any system; however, it requires each component to be individually correct, before the system as a whole can be properly evaluated. Due to the embedded nature of a KBS, the system evaluation is beyond the scope of this work. As an initial step toward the goal of overall system evaluation, our work presents a methodology for the evaluation of the knowledge base.

The next section clarifies the distinction between validation and verification. Following this, Section 2.2 details the rule-based approach to V&V and Section 2.3 the neural networks' approach. Finally, four significant issues regarding validation are defined and discussed.

2.1 Validation vs. Verification

The term "evaluation" has a nice feel, however, it lacks a concrete definition as far as it applies to knowledge bases. While MYCIN's developers preferred terms such as "debugging" and "evaluation" to discuss their testing and correction procedures [6], the task deserves more precise definitions. The general process of testing, evaluating, and correcting any software is termed "verification and validation" (V&V).[1] As they relate to knowledge bases, verification is the process

[1] Although [19] establishes a new nomenclature for the V&V of KBSs:"performance and quality"; our work uses the standard "verification and validation" for clarity.

of ensuring the knowledge base was built right, while the validation process ensures the right knowledge base was built [36].[2]

Verification is concerned with comparing the knowledge base against its specifications [17, 19]. This process ensures that the transfer of knowledge from human to machine-understandable form, does not violate any constraints of the knowledge representation. In practice, the knowledge representation may not have any intrinsic constraints; however, the inferencing mechanism that reasons over this knowledge representation will probably have constraints. These inferencing constraints are usually mapped onto the knowledge representation to ensure that a verified knowledge base can be inferenced over without causing an error.

Validation, on the other hand, is concerned with ensuring that the knowledge in the knowledge-base, when inferenced over, provides acceptable responses. This "sine qua non" [3] of knowledge-base evaluation can be defined as the comparison of the knowledge base against its requirements [19]. Both processes include the testing for and correction of their specific types of errors.

2.2 Rule-Based V&V Approaches

Approaches to the verification and validation of rule-based knowledge bases are numerous [20, 52, 33, 34, 4, 32, 50]. While many of these sources claim to provide methods and techniques for verification and validation, nearly all of them are solely concerned with verification. This preponderance of verification literature centers around a standard list of errors, developed through the years, against which rule-bases are to be checked. This list generally includes [17]:

1. redundant rules
2. conflicting rules
3. subsumed rules
4. circular rules
5. unnecessary IF conditions

[2] Since validation and verification are applicable to any software system, the typical phrasing is "building the right system" and "building the system right". However, the phrasing is slightly changed here to avoid confusion in light of the distinction between the overall knowledge-based system and the knowledge base itself.

6. dead-end rules
7. missing rules
8. unreachable rules

In this list, the top five items pertain to the *consistency* of the rule-base and the last three to its *completeness.* Generally, "rule-based verification" has been transformed into "consistency and completeness" [20, 6, 17].[3]

The historical dominance of rule-based systems has extended this "verification = completeness + consistency" concept to knowledge-based systems in general. The reason this extension has proceeded unchecked is due to the lack of specifications surrounding knowledge-base development. As mentioned previously, the specifications of a knowledge-base are actually the constraints of the inferencing method. It follows, then, that the main thrust in knowledge-base verification for all KBSs has actually evolved from the constraints of rule-base inferencing.[4] In other words, the push towards "completeness and consistency" as the goal of verification is an artifact of rule-based inferencing mechanisms, and not necessarily a requirement for all knowledge-base verifications. Any verification of a knowledge base with an inferencing mechanism able to accommodate incompleteness renders the "completeness" constraint moot.

Turning to validation, many authors recognize the need to validate the knowledge base [6, 20]; however, published results of actual validations are few. The three validations that MYCIN underwent are frequently cited as benchmarks in the evolution of knowledge-based *system* validation [6]; however, ***knowledge-base*** validation lacks historical benchmarks save TEIRESIAS [11]. The act of "knowledge-base debugging" [6] is accomplished on every KBS, but for some reason rarely defined or detailed. Viewed from the definition mentioned above, knowledge-base validation has historically

[3] As empirical evidence, the title of chapter eight in the Buchanan and Shortliffe book [6] is "Completeness and Consistency in a Rule-Based System".

[4] Note: At this time, little literature exists on the verification or validation of other than rule-based knowledge-bases; however, many authors claim their techniques apply to all KBSs.

lacked requirements [18]. Consequently, the techniques of knowledge-base validation have been left open for interpretation.[5]

2.3 Neural Networks V&V

As knowledge-based systems, neural networks [41, 42, 23] have a unique approach to verification and validation.[6] In applying the definition of verification, neural networks do not have any specifications regarding their "knowledge-base". Since their "inferencing method" propagates values along the connections without any reasoning *per se*, there are no constraints on the arrangement of information internal to the neural net.[7] Once the net is created, it can be considered verified. Validation, however, is not as trivial.

Neural network *systems* can be validated in much the same way as any other KBS; specifically, the techniques used in the formal validation of MYCIN apply. The net itself, metaphorically the knowledge-base, can also be validated. In fact, the training of the net (which is akin to knowledge acquisition) amounts to a continuous validation process. During this process, the net is presented with a set of inputs and a set of desired outputs. The input is then propagated through the net and its output is compared with the desired output. Any error in output, even if the solution was correct, is propagated backwards through the network adjusting the weights in the direction of the desired output.[8] After each pass of the training data through the network, an accuracy percentage is calculated by passing all the training data through and determining what percentage produces the desired output. This entire process is repeated many times until the overall accuracy

[5] While it may seem obvious that to validate ("debug") a rule-base, one merely needs to submit test cases and, if a "wrong" answer results, fix the incorrect rule(s); finding the incorrect rule(s) amounts to an exercise of the credit-assignment problem [9].

[6] Frequently excluded from discussion, neural networks are considered to be a knowledge-based system [16], and are included here due to their relationship to belief networks.

[7] In fact, the arrangement of information internal to a neural network is of little consequence to the workings of the net. Currently, research is being done to map the internals of neural networks into human-understandable rules, so that the principles of the internal workings can be better understood [28].

[8] This description portrays a "back-propagation" training algorithm, which is the most common type of training.

of the network is as high as possible. At the successful conclusion of this training, the neural net is considered validated for its training set.

Depending on the amount of available training data, different procedures are followed for the selection of training sets and testing sets. If a large amount of data is available, the data may be partitioned into distinctly separate testing and training sets. Each set must be a proper statistical sample of the entire problem-space, ensuring enough data is available to train properly and to test thoroughly. Typically, when the available data is limited, all of it is needed for proper training. Using all the data to train with, however, can result in overtraining.[9] Instead, a technique called "Hold-one-out" [29, 14, 40] can be used to train the net. This technique prevents overtraining, allows the use of the entire data set (although not on any one pass), and guarantees a statistically accurate metric for measuring accuracy.

2.4 Validation Issues

Four significant validation issues were defined by [36] and detailed in [16]: What to Validate, When to Validate, Which Methodology and What Criteria.

What to Validate: Generally, validation will examine either the final results of a query, the intermediate results of a query, or some combination of the two. As [16] points out, the final results are obviously the main concern; however, intermediate results (the "means to an end") can provide insight into the structure of the knowledge.

When to Validate: This topic was largely debated in the past between two general schools of thought. One side believed that validation could not be effective until the knowledge base was complete, because only then would it contain enough information to provide any reliable validation measurements. The other side, which eventually became more accepted, believed that validation was an integral part of the development of the knowledge base, and that it should

[9] This phenomena produces a neural net that learns the training data and only the training data; i.e. it fails to generalize.

be accomplished regularly as the amount of knowledge increases. While the latter approach is more favored, the former approach has implications for knowledge-based *systems* validation. It seems reasonable to have a system-wide evaluation at the conclusion of system development; while component validation, specifically the knowledge base, can be accomplished iteratively.

Which Methodology: This topic has generated a plethora of approaches for rule-based systems, some of which are cited in Section 2.2. [16] mentions five general approaches: informal validation, validation by testing, field tests, subsystem validation, and sensitivity analysis. However, in describing these "different" methodologies, it is apparent that a key concept is overlooked: testability. The validation process is supposed to measure a knowledge base against its requirements, this implies that the requirements must be testable. In other words, all five methods really describe different *sources* for the "question/evidence" portion of the requirements, not different validation methodologies. Currently, the only feasible method of applying requirements to a knowledge base is validation by testing.

What Criteria: Building on the methodology discussion above, this component of validation is the metric used to gauge the success of the knowledge base during testing. Many different criteria can be generated for a single set of requirements. Simple right/wrong percentages give one metric, while having a scale of how "bad" an answer is could provide another. Criteria seem only to be limited by the ability to measure features of the requirements and knowledge base.[10]

As is apparent by the above discussion, current validation techniques are largely based on the specific type of knowledge base being validated. Since this slant will continue until a universal knowledge representation is agreed upon, the evolution and definition of the Bayesian Knowledge Base is important to understanding the methodology detailed in Chapter IV. The next chapter provides the needed material to understand the knowledge base of choice for our work.

[10]It should be noted that [16] does not isolate metrics as the focus of this validation issue. Similar to the methodology discussion, different sources for the "answer" portion of the requirements (known results, human performance, theoretic possibility) are intermixed.

III. Knowledge Representation

3.1 Uncertainty

One of the difficulties traditional knowledge representations have is handling uncertainty; something humans accomplish regularly. For example, if you were told, "When it rains, the sidewalk gets wet" you would understand the meaning. Similarly, a KBS utilizing IF-THEN rules could be given: "IF raining = true THEN sidewalk-surface = wet". The uncertainty problem occurs when it is raining, but the sidewalk is not wet. A human may observe this apparent contradiction and remark "There's an exception to every rule." However, getting a KBS to accommodate this situation is not trivial. The IF-THEN rule could be modified to include exceptions, but at what point are enough exceptions listed? The main problem is that the original statement "When it rains, the sidewalk gets wet," is not completely true. It is true under *most* conditions, but not all. A knowledge representation that can incorporate uncertainty and accommodate knowledge from a variety of domains is essential in making a Knowledge-Based System that is both general and useful.

Development of the earliest systems showed that accommodating uncertainty was a must for making a KBS that could function well in diverse domains [6]. Currently many schemes of accounting for uncertainty in a knowledge representation exist, e.g. probabilities [37], certainty factors [6], Dempster-Shafer theory [12, 48], and fuzzy logic [51] to name a few. While some of these methods have been demonstrated in specific domains, probability theory, with its well-established mathematical foundation, provides sound and consistent knowledge representations for many domains [37, 44].

3.2 Bayesian Networks

One of the more popular knowledge representations accommodating uncertainty is the Bayesian Network [37, 7, 15, 26]. This knowledge representation models the probabilistic dependencies be-

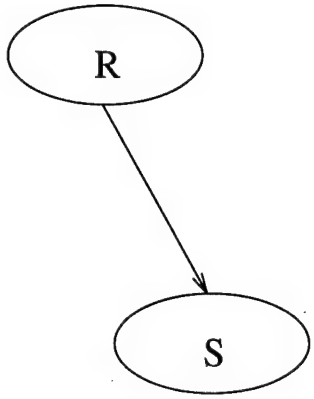


Figure 3.1 **Bayesian Network.** The Bayesian Network corresponding to the rule: "IF raining = true THEN sidewalk-surface = wet with probability of 0.9". Here "R" and "S" correspond to "raining" and "sidewalk-surface" respectively.

tween discrete objects, termed random variables (RVs). Resuming the above example, and adding probabilities to modify the rule yields: "IF raining = true THEN sidewalk-surface = wet with a probability of 0.9."[1] The more general approach taken by Bayesian Networks eliminates the IF-THEN structure, and associates "raining = true", "sidewalk-surface = wet", and "0.9." Bayesian probabilities[2] (subjective probabilities) allow this type of association to be made. To illustrate:

$$P(A = a \mid B = b) = p$$

means p is the probability that $A = a$ given that $B = b$. The example now becomes:

$$P(\text{sidewalk-surface} = \text{wet} \mid \text{raining} = \text{true}) = 0.9.$$

Graphically, this relationship is shown in Figure 3.1.

[1] For clarification, here the RV's are "raining" and "sidewalk-surface".

[2] These probabilities are based on Bayes' rule: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$. For discussion of Bayes' rule and Bayesian Networks see [35, 8, 37].

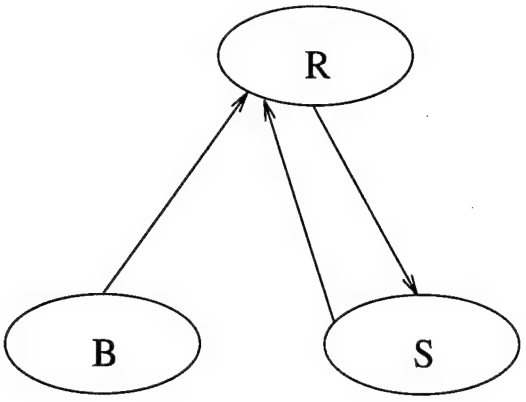


Figure 3.2 **Improper Bayesian Network.** The Bayesian Network from Figure 3.1 with this rule added: "IF sidewalk-surface = dry AND building-shelters-sidewalk = false THEN raining = false with a probability of 0.8".

There are, however, restrictions on the arrangement of information in a Bayesian network due to theoretical limitations [44]. Continuing the example, the following rule needs to be added to the knowledge base: "IF sidewalk-surface = dry AND building-shelters-sidewalk = false THEN raining = false with a probability of 0.8", see Figure 3.2.

Such "circular dependencies" are fairly common to humans, however classic Bayesian networks prohibit them; limiting the flexibility and versatility of the knowledge representation. Further limiting usefulness, an exhaustive list of RV-states, as well as every dependent and prior probability, must be supplied to the Bayesian network before inferencing is possible.[3] Often, these numerous (combinatoric) probabilities are not known, and do not make any sense to a human; defining them becomes a major stumbling block in knowledge acquisition.

[3]There is some current work on inferencing without a complete probability specification, [38, 39]; however, this work does not accommodate finer knowledge resolution nor allow for incomplete RV-state specification. This approach, while valid, is orthogonal to the work presented here which directly addresses those limitations of Bayesian Networks. (See Section 3.3)

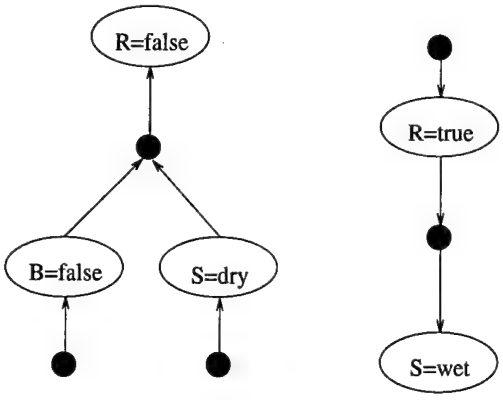


Figure 3.3 **Bayesian Knowledge Base.** The Bayesian Knowledge Base representing the two rules from Figure 3.2. Notice that the BKB can accommodate RV level cyclicity–something a Bayesian Network cannot.

3.3 Bayesian Knowledge Base

These limitations have been overcome by the Bayesian Knowledge Base (BKB) [44]. The BKB is a generalization of the classic Bayesian network capable of incorporating more detailed probabilistic dependencies and incomplete knowledge of RV-states and dependencies.[4] The BKB subsumes Bayesian networks, and remains firmly rooted in probability theory.

The two key differences between Bayesian Networks and BKBs are: knowledge granularity, and knowledge completeness. The BKB records probabilistic dependencies at the instance (state) level, instead of the RV level; i.e. the smallest piece of knowledge in a BKB is an RV-instance (raining = true), as opposed to the Bayesian Networks' RV (raining). Figure 3.3 shows the BKB needed to represent the two rules present in Figure 3.2.

A BKB is a bipartite directed graph composed of two distinct types of nodes. The lettered ovals, termed "instantiation nodes" or "I-nodes", represent specific RV-instances. The filled circles, "support nodes" or "S-nodes", each represent one probabilistic dependency relationship, and con-

[4]Exactly which probabilities and which RV-states are required by any BKB is dependent upon its validation, see Chapter IV.

tain the probability associated with that relationship. The BKB has no requirement for exhaustive RV-state lists or complete probability specifications. This allows it to accommodate incomplete as well as uncertain knowledge.

3.4 Knowledge Acquisition and Consistency

Any knowledge representation is of little use without a means of knowledge acquisition. The MACK tool presented in [46, 2] is the knowledge acquisition device for the Bayesian Knowledge Base. MACK supports an incremental knowledge acquisition philosophy, allowing the knowledge engineer freedom in constructing the knowledge base. This tool uses linguistic parameters to help overcome the typical difficulties in defining exact probabilities for a knowledge base. Additionally, it provides a mechanism for handling temporal information, and guarantees knowledge base consistency.

This consistency is actually the verification of the knowledge base. MACK compares the structure of the knowledge base against a set of constraints (specifications) that ensure the inferencing method is able to perform correctly. BVAL guarantees that the knowledge base will remain verified during its operation.

3.5 Inferencing

The purpose of any Knowledge-Based System is to provide information (diagnosis, advice, analysis, etc.) to a user. The process of deriving this "information", termed inferencing, is of central importance to any KBS. The two traditional methods of inferencing over Bayesian Networks, Belief Updating and Belief Revision, apply to BKBs and are explained in the following sections. Additionally, a new method of inferencing unique to BKBs is also discussed.

3.5.1 Belief Updating. Belief Updating [37] seeks to answer the question "What is the probability of X, given the evidence?", where X is a single element[5] of the knowledge base and "evidence" is a set of elements known to be either true or false.[6] This, and all types of inference over general Bayesian Networks, is NP-hard [10],[7] and as such is only useful in situations where the actual *probability* of a certain element is important. While these situations do occur, of greater use over a variety of domains is Belief Revision.

3.5.2 Belief Revision. Belief Revision [37, 44, 47] finds the "most probable explanation" (MPE) for the given evidence based on the optimal joint probability of the random variables. This is actually the most likely state of all random variables in the knowledge base; or more informally, the most likely state of the world. This type of inferencing is most useful in explanatory/diagnostic domains where the exact probability of any particular element is not as useful as the inclusion of an element. For example, in a medical domain (diagnostic) a patient may have chest pain (evidence). Inferencing with Belief Revision will result in every RV being assigned a state (instantiation). One of the RV-states in the MPE may be "congestive-heart-disease = true". The presence of this element is of more value to the user of the system and/or the patient, than is the exact probability that the patient has congestive heart disease (Belief Updating). In fact, the probability of having a given disease may be fairly small on an absolute scale; however, the fact that the disease is a part of the "most probable state of the patient" is significant.

While this inferencing method seems to provide the desired information in its applicable domains, a problem occurs when it is used on an incomplete knowledge base (BKB). The problem has two main roots: the "type" of information in the knowledge base, and the "globalness" of the

[5]For sake of generality and conciseness, "element" refers to the smallest unit of knowledge in the knowledge base; i.e. an RV for Bayesian Networks, an RV-instance for BKBs.

[6]This does not mean that the element itself is bi-valued or boolean, instead it asserts or denies the presence of the element in observations made before inferencing.

[7]There is a special class of Bayesian Networks called "singly connected networks" or "polytrees" that can be inferenced in polynomial time [37, 49]; however, by definition they are not the general case [8].

inferencing solution. While these root problems seem disparate, they actually combine to address the broader topic of relevancy.

"Type" of information is best described by way of example. Remaining in the medical domain, the knowledge acquisition process will probably acquire only "positive" information; i.e. information that can be used to conclude the presence of some disease. "IF chest-pain = true THEN congestive-heart-disease = true" (positive information) is more likely to appear in the knowledge base than an exhaustive listing of exactly what will cause "congestive-heart-disease = false". It seems reasonable that the knowledge engineer, when given the freedom to supply incomplete information, will supply only the "important" or relevant information. From here, it is easy to see how the BKB would contain many "disease-X = true" instances, and few "disease-X = false" instances.

As mentioned above, Belief Revision results in every RV being assigned its most likely state with respect to the evidence. This "global" assignment of every RV will result in the most likely assignment for all "disease-X" RVs. However, since many of those RVs have only one state in the knowledge base, their state assignment is a foregone conclusion. What this means to the user is every disease with only one state, or all positive states, will be present in every solution; e.g. no matter the evidence, the patient's most likely state will include (say) flu, cold, pneumonia, meningitis, ulcer, congestive heart disease, etc..

A possible solution to this problem is to make a single "disease" RV, that has many states corresponding to flu, cold, etc.. This method, however, has problems of its own. It cannot accommodate positive and negative information for a single disease, it cannot account for one disease being a symptom of another disease (e.g. pneumonia is a symptom of AIDS), and it only allows one disease to be present in any solution (since a random variable cannot be in two states simultaneously). Each of these problems is as unacceptable as the original problem of having many diseases every time.

Another possible solution is to utilize Belief Updating instead of Belief Revision. To make use of Belief Updating, the inferencing method would have to check every disease RV and rank the diseases according to probability. Then some (arbitrary) decision would have to be made on how many of the top diseases to include in the solution. Besides the possibly arbitrary decision just mentioned, this method takes an extraordinary amount of time to arrive at its solution since each separate Belief Updating pass is NP-hard. Additionally, there is an implicit assumption that the disease RVs are somehow known *a priori*. While the knowledge acquisition methodology could be changed to accommodate this restriction, this technique provides no information about how other elements relate to the solution. Thus, an interactive session attempting to confirm/deny the conclusion with additional information is not possible. Again, this potential solution results in limitations equaling the original problem.

The ideal solution is to inference over (assign states to) only those random variables that are relevant to the evidence.

3.5.3 Belief Extension. Belief Extension [45] is a new inferencing method that only works on BKBs, and is the inferencing method utilized for our work. This method is related to Belief Revision and was developed to overcome some of the problems mentioned above regarding the use of Belief Revision in an incomplete environment.[8] This method inferences over only those instances and RVs that are determined to be probabilistically relevant to the evidence. In general terms, this technique can be thought of as performing Belief Revision over only the information that directly pertains to, or derives from the evidence; i.e. no RV is forced into a state unless it is relevant to the evidence. This technique tends to minimize the number of extraneous RV-states included in the solution due to the "globalness" of Belief Revision.

[8]However, this method is not the ideal solution referred to above; the "true" problem of relevancy is extremely difficult and beyond the scope of this effort.

A side effect of this technique, in its purest form, is the absence of any determination about the state of the world beyond the probabilistically relevant region. Continuing with the medical example, this could mean that the solution for a given set of evidence does not contain any disease instantiations. While a solution of this form is precise, it fails to include information that is important to the user; e.g. the disease most likely present given the evidence. Put another way, pure Belief Extension will not speculate beyond the facts it can substantiate. To correct for this, Belief Extension can utilize a "look-beyond factor". This factor causes the inferencing method to speculate beyond its relevant region by some amount. This speculation actually involves assigning a state to every RV reachable by "looking beyond" the relevant region a "factor" of times. For an example see Figure 3.4. Here the relevant region is extended by a look-beyond factor of one. In simpler terms, inferencing results in an assessment of the most probable state of the partial world defined by the evidence and the look-beyond factor.[9]

Inferencing over an incomplete knowledge base poses an additional problem pointed out in [46, 2]: no matter what the inferencing technique, it is possible to be unable to compute a solution. This occurs due to a conflict between strict adherence to probabilistic semantics and a lack of information. To overcome this problem, a type of default reasoning is invoked. This method of "default as negation" causes an element to be assumed if it is necessary in order to compute a solution. No assumptions are ever made if there is any way to get a solution without them.

The end result is an inferencing method that functions well over an incomplete knowledge base.

3.6 Solution vs. Answer

The previous section mentioned, but never defined, the "solution" that results from inferencing. The solution, either from belief revision or belief extension, is the set of elements found to

[9]The look beyond factor can be determined by the user; however, this factor can also be determined by an application of case-based reasoning to the test cases of the validation process. See Section 6.6.

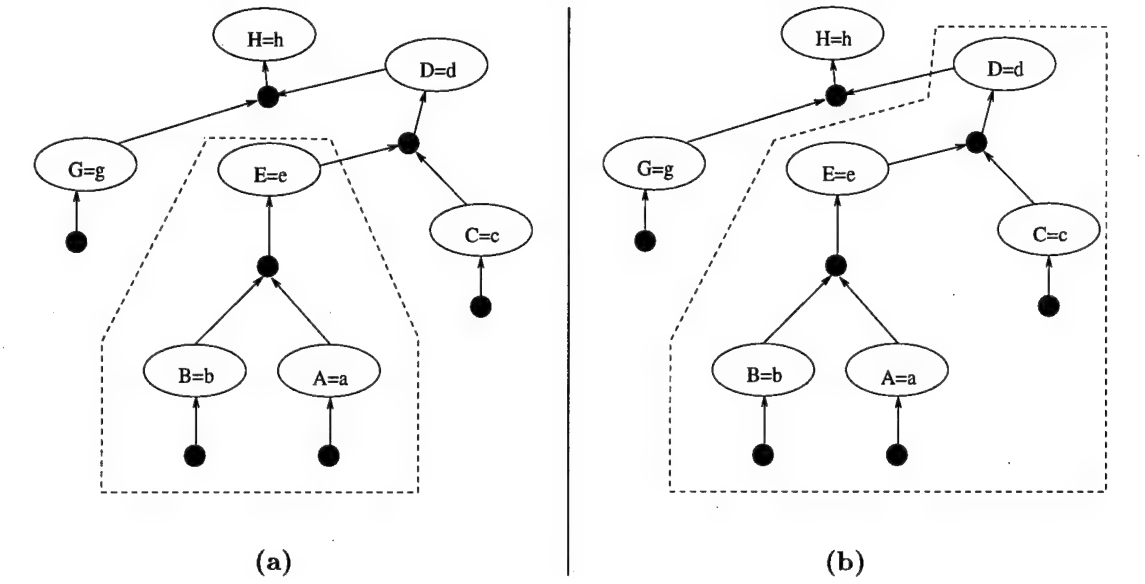


Figure 3.4 **Look-Beyond Factor.** In these figures, E = e is the evidence. Part (a) shows a BKB and the probabilistically relevant region encompassed in a dashed line. Part (b) shows the same BKB with a look-beyond factor of 1 applied to the relevant region.

compose the most likely (partial) state of the world. This set forms a subgraph of the knowledge base and usually has paths that include the evidence.[10] This solution provides important information for validation (see Chapter IV), and presents a problem for providing an answer to the user.

To clarify, "solution" refers to the output of the inferencing mechanism, while "answer" refers to that portion of the solution presented to the user. The need for this distinction is apparent when dealing with a Belief Revision solution, although just as necessary with Belief Extension. Recall that a Belief Revision solution contains the state (possibly assumed) of every RV in the knowledge base. Returning to the medical example and bypassing the problems discussed previously, the evidence is "fever = true", and the solution now contains "cold = true", "shivering = true", "coughing = true", "sore-throat = true", etc. What is the *answer*?

[10] With the use of assumed nodes it is possible to get a subgraph that does not have any paths, just isolated nodes.

One approach to this "interpretation" problem is to not interpret the solution at all but just display the entire solution to the user. It is easy to see, however, that such an approach quickly degrades in usefulness as the size of the knowledge base increases. Forcing the user to examine a large number of RV-states, looking for the one (few) that provide useful information is not effective. The real crux of this problem is somehow deciding what element(s) of the knowledge base are relevant to the solution given the evidence. The following approach attempts to address this problem by extending the knowledge acquisition process and adding functionality to the user interface.[11]

This approach involves having the expert/knowledge engineer provide additional information during knowledge acquisition. This "additional information" will be the "type" or "category" of the random variable. For example, the RV "fever" might be type "symptom", the RV "cold" might be type "disease", and the RV "pneumonia" might be type "symptom" and type "disease". The exact type label(s) should be determined by the expert/knowledge engineer, since they are the source of information contained in the knowledge base.

Now that this additional information is in the knowledge base, it must provide some functionality. The user interface can utilize this information to filter the solution as reported by the inferencing mechanism. For example, if "cold = true", "shivering = true", "coughing = true", and "sore-throat = true" compose the solution, and "cold" is of type "disease" while the others are all of type "symptom", the solution can be filtered for "disease" RVs resulting in the *answer* "cold = true".

In deciding what types to report, the expert/knowledge engineer could provide a list of "reportable types" as a default filter for each query based on the purpose of the knowledge base; e.g. all RVs of type "disease" are included in the answer by default. The user may also specify

[11] While it would be possible to use some type of case-based reasoning in conjunction with the validation process (similar to the look-beyond) to arrive at an approximation of the answer, that approach is limited without the additions mentioned here.

types to include or exclude from the answer, giving them the flexibility to tailor the output without changing the knowledge.

3.7 Moving On

Summarizing this chapter, the need to include uncertainty within a knowledge representation has led to probabilities. Probabilities evolved into Bayesian Networks and Bayesian Network's limitations produced Bayesian Knowledge Bases. Having a knowledge representation led to acquiring knowledge. A populated knowledge base is useless without some means of extracting information; enter inferencing. Belief Updating provides the exact probability of a specific element, Belief Revision provides the most probable state of the world. A third inferencing technique, Belief Extension, addresses some the practical difficulties of Belief Revision in providing the most probable relevant state of the world. Finally, the solutions returned from inferencing are filtered to provide the user with an answer. Apparently we have gone full circle, from knowledge representation to answer; however, one crucial point has been carefully ignored until now: Is the solution *correct*?

IV. Bayesian Knowledge Base Validation

Ideally, the process that ensures an acceptable solution is provided for every query would be completely automated. However, since BKBs incorporate incomplete knowledge, this is not possible. Incomplete knowledge produces errors that can only be corrected by modifying the amount of knowledge in the knowledge base. Automatically performing this correction requires an awareness of the knowledge that is either missing or extraneous. Having an awareness of what knowledge is needed to automatically correct an incomplete-knowledge error, implies that the system already contains the knowledge needed to fix the error. Completing this circular argument, if the system already knows what knowledge is missing or extraneous, why would an incomplete-knowledge error ever occur? Clearly, the allowance of incomplete knowledge requires the user[1] to interact with the knowledge base during validation. Keeping the ideal in mind, the goal of BVAL is to minimize the interaction required.

BVAL's approach to validating Bayesian Knowledge Bases achieves this goal by combining both rule-based and neural net validation techniques. The rule-based approach emphasizes completeness and correctness while only providing the user with assistance in locating the incomplete/incorrect knowledge. The neural net approach assumes its knowledge is complete[2] and automatically reinforces correct knowledge. Since BKBs will have incomplete and incorrect knowledge, BVAL combines these techniques to provide automatic correction whenever possible and user assistance otherwise.

This chapter is devoted to describing BVAL's methodology and the reasoning behind the methodology. The first section illustrates BVAL's approach in relation to the four main validation issues described in Chapter II. Next, the methodology is described in more detail from a big-

[1]The term "user" refers to the individual(s) performing the validation; this may include an expert(s), knowledge engineer, end user, or a combination of all three.

[2]If enough data exists, the testing and training sets can be statistically analyzed for completeness; however, there is never any guarantee that the data correctly samples the domain space.

picture point of view. Finally, the last sections detail the three important areas of BVAL: test cases, reinforcement, and incompleteness.

4.1 Validation Issues

Before detailing BVAL's methodology, it is important to understand its overall approach in general terms. The PESKI system (Appendix A) will be released as a KBS *shell*, a fully functional system without any knowledge. Since each user will need to validate their individual knowledge base, BVAL was developed for inclusion in the system shell. Because of this arrangement, BVAL is designed to directly compare the knowledge base against its requirements. The requirements of the knowledge base are a set of test cases, where each test case has evidence and an expected answer. This requirements set is similar to the training/testing sets of neural networks. BVAL is designed to be useful over, and an integral part of, the whole knowledge base lifecycle.

Having a rough overview of BVAL's approach, a discussion of the four significant validation issues (Section 2.4) as they pertain to BVAL is now possible.

What to validate: Since validation is concerned with ensuring an acceptable solution is provided for every query, it seems obvious that the final result, the solution, is being validated. The intermediate results, while a part of the solution, are immaterial if the solution is correct. If an intermediate result is to be included in the solution, then it should be part of the expected answer for that test case. BVAL's methodology will then ensure that this intermediate result is a portion of the solution.

When to validate: BVAL is designed to be used at all stages of the knowledge base lifecycle. Since knowledge can be acquired iteratively, it makes sense that it should be validated iteratively. It is expected that as the knowledge base is developed, it will be constantly validated to ensure that the knowledge acquisition process is working properly. This type of validation also lends a sense of confidence to the user, as they can see correct solutions being produced by the knowledge base.

Validating the final knowledge base will ensure that the knowledge is as complete as needed and as correct as possible given the requirements. As the knowledge base enters the maintenance portion of its lifecycle, additional benefits of BVAL's methodology are seen. Reports of incorrect solutions can be directly incorporated into the requirements of the knowledge base. The knowledge base can then be re-validated with the new requirements; BVAL will automatically correct the knowledge base, or guide the user toward areas of unacceptable incompleteness.

Which Methodology: As mentioned in Section 2.4, the only feasible methodology currently available that can formally compare a knowledge base against its requirements is validation by testing. BVAL uses this methodology throughout its lifecycle.

What Criteria: Currently, the measurement criteria used by BVAL is similar to that used by neural networks; a simple percentage, calculated by dividing the number of correct test cases by the total number of test cases. This criteria will give an accurate measurement of the knowledge base at any time during its development. The exact accuracy percentage desired is defined by the user; however, it is theoretically possible to make any BKB 100% accurate, with respect to a given set of requirements, by inputing enough knowledge to account for all possible combinations of evidence (i.e. a complete knowledge base). The intent of BKBs, however, is to allow for incomplete knowledge so that the same accuracy percentage can be achieved with less input from the user and less storage requirements for the knowledge base. Furthermore, recall that the entire KBS, should also be validated and verified. This system-wide KBS validation should include some metric relating to the expected long term reliability of the knowledge base; however, at this time that metric has not been formulated.

4.2 BVAL's Methodology

The approach taken by BVAL to validate a single test case is best described by a list of actions and tests followed by discussions of each alternative.

1. A test case containing evidence and the expected answer are submitted.
2. A query is run, returning a solution.
3. If the solution is correct (contains the expected answer) and no assumptions were made, that test case is considered validated, and the next test case is processed.
4. If the solution was correct, but contained assumptions, then the knowledge base is incomplete and needs to be corrected. (Note that the term "incomplete", used in this fashion with reference to a BKB, means "unacceptably incomplete in a specific area highlighted by the test case". This is not meant to imply that the knowledge base should be made "complete".)
5. Otherwise, the solution was incorrect, making the knowledge base a candidate for reinforcement learning on this test case.

Once a candidate for reinforcement learning is identified, a second query is submitted with the test case evidence and the expected answer both provided to the inference engine as "evidence". This forces the inference engine to produce a solution containing the expected answer. If this solution, which is known to be correct, contains assumptions, then the knowledge base is incomplete and needs to be corrected. Otherwise, the knowledge base just needs to be reinforced to accommodate this test case. Once the reinforcement procedure is complete, the validation proceeds to the next test case.

If the knowledge base was found to be incomplete, the user is given the option of bypassing this test case, or attempting to correct the knowledge base. If the user wishes to correct the knowledge base, they are provided different means of viewing the problematic knowledge in order to efficiently deduce the remedy. Once the user has decided upon a remedy, they are allowed to edit the knowledge base using the familiar MACK tools. Upon completion of editing, the original test case is run again. This second check ensures the corrections made actually work–providing immediate feedback to the user. Since this second pass is handled like any other test case, if the knowledge base is still incomplete, the user can continue to edit. Additionally, this format allows the user to make iterative corrections to the knowledge base without altering the test suite.

Once every test case in the suite has been processed, all cases are passed through again, this time for metric purposes (each case is either correct or incorrect with no additional processing). The percent correct is considered the current accuracy level of the knowledge base. Since the

reinforcement learning caused by one test case can impact the validation status of another test case, the entire test suite is run again (as the first time with reinforcement and interaction as necessary) with the test cases in a random order. The constant changing of the ordering of test cases is a technique borrowed from neural networks and is important here since order can affect validation.

After going through this process a number of times, the knowledge base should start to show a fairly constant level of accuracy. If this level of accuracy is acceptable to the user, then the knowledge base is considered validated for this set of requirements. If the accuracy is not acceptable, then further analysis is warranted. After all of the test cases causing incomplete errors have been corrected, BVAL should be running without user interaction; i.e., reinforcement is the only corrective action being taken. At this point, if the knowledge base is not achieving an accuracy of 100%, then thrashing is occurring between test cases (See Section 4.4.3) indicating areas of incompleteness that need to be corrected. After the thrashing errors have been corrected, one final pass through the test suite should show an accuracy of 100%—indicating the knowledge base is validated against its requirements.

4.3 Test Cases

4.3.1 Application. BVAL uses the MACK tools the user is familiar with to perform test case solicitation. Here the user is being prompted for the evidence portion of the test case used for Figure 4.1.

```
Please enter the EVIDENCE
Please pick a component:

    1 – Acute Bronchitis
    2 – Chronic Bronchitis
    3 – Cough
    4 – CXR
    5 – History and Physical
    6 – Interstitial Fibrosis
    7 – Malignant Disease
```

8 – Obstructive Pulmonary Disease
9 – PFTs
10 – Pneumonia
11 – Sputum or Tissue Culture
0 – Abort
Choice: 3

Please pick an instantiation of Cough:

1 – Cough = TRUE
0 – Abort
Choice: 1

Please enter the EVIDENCE
Please pick a component:

1 – Acute Bronchitis
2 – Chronic Bronchitis
3 – Cough
4 – CXR
5 – History and Physical
6 – Interstitial Fibrosis
7 – Malignant Disease
8 – Obstructive Pulmonary Disease
9 – PFTs
10 – Pneumonia
11 – Sputum or Tissue Culture
0 – Abort
Choice: 0

Although this example shows only one RV-state (instantiation) being submitted as evidence; the answer and evidence can contain many elements. Following the entering of evidence, some calculations are performed to find the connected region of the knowledge base (Appendix B). The expected answer must be contained in this region; this quality being guaranteed during solicitation. Additionally, a check for potentially thrash-causing test cases is performed as each expected answer is submitted (Section 4.4.3).

Please enter the expected ANSWER
Please pick a component:

1 – Acute Bronchitis
2 – Chronic Bronchitis
3 – Cough
4 – CXR
5 – History and Physical

6 – Interstitial Fibrosis
7 – Malignant Disease
8 – Obstructive Pulmonary Disease
9 – PFTs
10 – Pneumonia
11 – Sputum or Tissue Culture
0 – Abort
Choice: 2

Please pick an instantiation of Chronic Bronchitis:

1 – Chronic Bronchitis = TRUE
0 – Abort
Choice: 1

Please enter the expected ANSWER
Please pick a component:

1 – Acute Bronchitis
2 – Chronic Bronchitis
3 – Cough
4 – CXR
5 – History and Physical
6 – Interstitial Fibrosis
7 – Malignant Disease
8 – Obstructive Pulmonary Disease
9 – PFTs
10 – Pneumonia
11 – Sputum or Tissue Culture
0 – Abort
Choice: 0

Following successful completion of test case solicitation, the validation process continues as outlined in Section 4.2. Details regarding the test cases are presented in the remaining sub-sections.

4.3.2 Numbers. Ideally, the test suite (requirements) would contain every possible query that could be submitted to the knowledge base; however, in all but the smallest of domains this is impossible. It is possible to generate all combinations of evidence automatically; however, the user must provide an expected answer for each combination to complete the requirements. Clearly, this approach is infeasible. Not only would a combinatoric number of test cases be generated, but a large subset of these cases would be meaningless and unrealistic for the domain. Barring such an exhaustive listing, the test suite should contain those cases representative of the entire domain.

Ensuring the requirements are representative of the domain is the same problem the neural network community must solve in relation to the testing and training sets used on neural nets. Unfortunately, as [40] points out, the neural network community cannot ever solve this problem. Even if a neural network is trained properly and passes all of its validation tests, there is still no guarantee that the network will ever solve a new problem correctly.[3] Extensive testing, if the data is available, provides some indication; however, all testing is based on the assumption that the available data correctly represents the problem domain.

Applying the lessons learned from the neural network community: data is good, more data is better. For neural networks this is not always true; data points corresponding to exceptions can detract from the accuracy of the net. For BKBs, however, if an exception should be accommodated by the knowledge base, it should be included in the test suite. This inclusion may result in an interactive session during validation, but BVAL will ensure that the exception is accounted for without detracting from the "non-exception" test cases. Bottom line, as many realistic test cases as possible should be included in the requirements.

4.3.3 Sources. Historical records, if available, are ready-made test cases and provide an excellent source of information about the domain. Experts can be recruited to prepare test cases; however, motivation can be difficult when the expert is busy. Additionally, another problem occurs when multiple experts are consulted: disagreement. This can sometimes be solved by putting the experts on a panel and having them agree on an answer. A more promising method, left for future development, is to have the panel rank answers. A test case would then consist of a set of evidence, and a rank ordered list of expected answers. Since the inferencing method can provide the second, third, etc., most probable solution, these answers can be validated in their rank position. This approach seems beneficial in giving depth to the validation, and providing a means of allowing

[3]This uncertainty is present because of the lack of complete knowledge about any domain. Aside from computer generated worlds, most domains occurring in the real world are subject to the whims of nature, which is inherently chaotic and unpredictable. Completely bounding a domain seems impossible to prove.

many experts to contribute to the test suite. Finally, since BVAL will be utilized throughout the lifecycle of the knowledge base, real-world cases that occur and are improperly solved should be included in the requirements. These "field test" cases serve to continually improve the utility and accuracy of the knowledge base.

4.4 Reinforcement Learning

4.4.1 Application. In this portion of its methodology, BVAL compares the incorrect solution generated by the evidence to the known correct solution generated by the evidence and the expected answer combined. Recall that these solutions are actually paths through the knowledge base, and only one state can be active for any RV in a solution. The two solutions' joint probabilities will be different, with the correct solution's probability less than or equal to the incorrect solution's. This known ordering of the solutions' probabilities is used to compute the "reduction factor". This factor is the percentage the incorrect solution's probability would have to be reduced to be lower than the correct solution's probability. Every probability (S-node) in the knowledge base, except the ones associated with the correct solution, are then reduced by this factor. After this reduction is applied, an up-scaling factor is applied to all probabilities in order to prevent precision errors.[4] Once this up-scale factor is applied, the reinforcement process is complete. If the same test case was re-run immediately, the correct solution would result.

Figure 4.1 shows the progression of changing probabilities during a reinforcement learning process (Section C.1 shows the entire BVAL session for this figure). This example is derived from a decision tree on respiratory disorders presented in [22]. Initially, the query is submitted with evidence of Cough = TRUE. (Table 4.1 shows the correspondence between the alphanumeric RV-states and their more descriptive terms.) This query returns the solution amounting to the left-hand

[4] While applying the reduction and up-scaling factors to the "entire knowledge base" as indicated is sufficient, it is not necessary. It is only necessary to apply these factors to the "connected region" of the knowledge base defined by the evidence. See Appendix B.

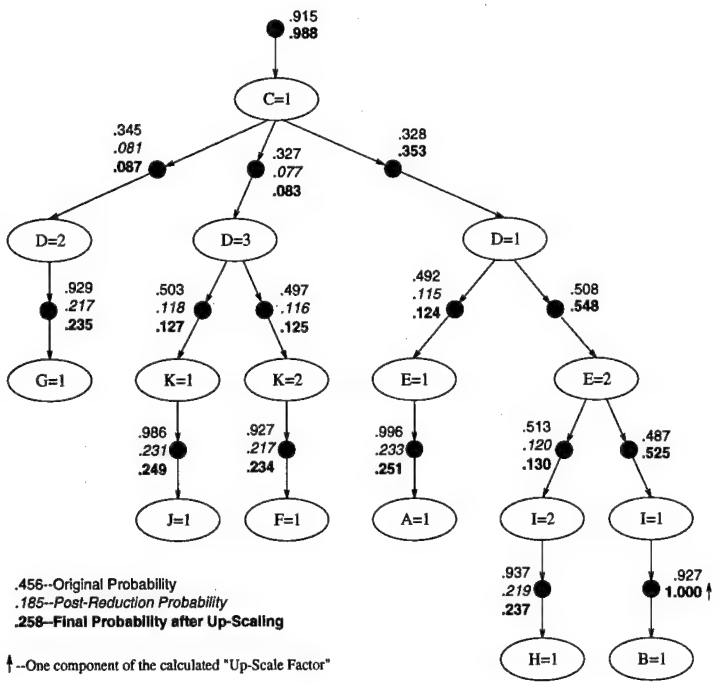


Figure 4.1 **Reinforcement Learning.** The probabilities change as reinforcement learning is applied to this knowledge base. Here the evidence was $C = 1$ and the expected answer was $B = 1$. Table 4.1 shows the translation of the alphanumeric RV-states to their more descriptive terms. (See Section C.1 for the transcript)

branch of the graph.[5] Since this solution does not contain the expected answer of $B = 1$, another query is submitted with $\{B = 1, C = 1\}$ as evidence. The solution from this second query is guaranteed to be correct, and since it does not contain any assumptions, the reinforcement process is started. After the reduction factor is applied, the up-scaling factor is derived, coincidentally from $B = 1$, then applied. The knowledge base ends up in the state denoted by the final probabilities. It is apparent that a re-submission of the initial query with $C = 1$ as the evidence would result in the correct answer.

[5]The full solution is {Cough = TRUE, CXR = Abnormal with Nodular Infiltrate, Malignant Disease = TRUE} or $\{C = 1, D = 2, G = 1\}$.

RV Name		RV State	
Alpha	Descriptive	Numeric	Descriptive
A	Acute Bronchitis	1	TRUE
B	Chronic Bronchitis	1	TRUE
C	Cough	1	TRUE
D	CXR (Chest X-Ray)	1	Normal
		2	Abnormal with Nodular Infiltrate
		3	Abnormal with Diffuse Infiltrate
E	History and Physical	1	Infection
		2	No Infection
F	Interstitial Fibrosis	1	TRUE
G	Malignant Disease	1	TRUE
H	Obstructive Pulmonary Disease	1	TRUE
I	PFTs (Pulmonary Function Tests)	1	Normal
		2	Abnormal
J	Pneumonia	1	TRUE
K	Sputum or Tissue Culture	1	Positive
		2	Negative

Table 4.1 **Mapping alphanumeric names to descriptions.** This table shows the correspondence between descriptive names and and alphanumeric names in the example associated with Figure 4.1.

4.4.1.1 Why percentages? Using percentages to alter probabilities keeps all the magnitudes in the same relative position. For example, $P = 0.6$ and $Q = 0.3$ are reduced by 50% resulting in $P = 0.3$ and $Q = 0.15$. Even after the reduction, P is still double Q; the relative magnitudes are identical. Reducing the original probabilities by a constant, say 0.4, results in $P = 0.2$ and $Q = -0.1$. Not only are P and Q now of different relative magnitudes, but Q is negative; a practical impossibility and a consistency violation. If some error checking is done to prevent $Q \leq 0$, a side-effect results in some initially different probabilities being equal after error checking and correction. This side-effect actually changes the knowledge, an unacceptable result.

4.4.1.2 Why compute the reduction factor that way? In Belief Extension, it is possible to have solutions with different numbers of RV-states contributing to the joint probability of each solution (See Figure 4.1).[6] Additionally, it is not known how many solutions are between the

[6] Belief Revision has the same number of RV-states in every solution since it requires every RV to be instantiated in any solution.

initial incorrect solution and the correct solution. It is possible that the two solutions are the first and second most probable; it is also possible that they are the first and tenth (or whatever) most probable. Combining these two facts, BVAL needs to guarantee that the correct solution will be the most probable after application of the reduction factor. To guarantee this quality, every individual probability is reduced (in percentage) by the amount that the overall joint probability needed to be reduced.[7] This technique guarantees the original solution and every intermediate solution, no matter how many RV-states might contribute to them, are less than the correct solution.

It should be noted that the ordering of the probabilities is guaranteed to be true since the inferencing method (either Belief Revision or Belief Extension) searches for the globally optimal solution. The first solution generated, in this case the incorrect solution, is the globally optimal solution—it has the largest probability. Any other solution based on the same evidence, or a superset thereof, must not have a probability greater than the original (since it would have been found the first time).

4.4.1.3 Why a single reduction? It is possible to use a stepping percentage instead of a single reduction factor; however, each time a step is applied, the query would have to be run again until the desired learning was accomplished. Since each inferencing pass is NP-hard, this iterative technique would take too long.

4.4.1.4 Why reduction and up-scaling? If the correct solution is increased in probability, as one may infer from the "reinforcement" label, it is possible to violate the consistency of the knowledge base; i.e. a probability or group of probabilities could exceed 1.0. Since maintaining consistency is required of the validation process, reduction is the method of reinforcement.

Each time the knowledge base is reinforced, the average probability moves closer to zero. Put another way, if all the probabilities form a surface map, the entire surface migrates closer to zero

[7] Actually the reduction factor is slightly smaller than the exact ratio of correct probability to incorrect probability. The ratio is reduced by a small percentage to ensure the correct solution is the most probable, not one of the most probable.

with each reduction pass. If computers were continuous machines, this would not be a concern; however, computers are discrete and have a fixed precision. This fact results in the possibility that, given enough reinforcement, most all probabilities would eventually be zero due to precision error. This difficulty mimics the problems of using constants instead of percentages. To avoid this situation, each time reinforcement is applied, the knowledge base is also scaled up as high as possible. Returning to the probability surface map, this is akin to elevating the entire surface as close to 1.0 as possible. To accomplish this, all compatible probabilities [46, 2] are summed and the largest sum is set to 1.0, by a percentage. This "up-scale" percentage is then applied to the knowledge base keeping every probability (including the newly reinforced ones) in their same relative positions.

4.4.2 Probabilities as weights. The reinforcement portion of BVAL's methodology freely manipulates probabilities; depriving them of any special status or significance. In fact, BVAL views the probabilities much as weights, where the inference engine is seeking the globally optimal largest weight. Although [46, 2] state "...the actual numeric value assigned to any given probabilities [sic] is not significant", many would question the meaning of any probability after its initial value is altered. To properly address this concern, a more philosophical discussion is warranted.

The crux of the concern many have with manipulating probabilities stems from the desire to attach absolute meaning to the joint probability of a solution. If such a meaning was attached, the knowledge base would have to be a duplication of the domain. For example, the relationship between "sidewalk-surface" and "raining" would have to be completely defined by "0.9". It does not seem likely that any knowledge base, including the mind of an expert, contains all the information necessary to perfectly represent a domain. A more appropriate interpretation of a knowledge base is that of a domain model not a domain duplicate. The knowledge base should model the domain accurately enough to arrive at correct solutions; not account for everything in the universe that could impact the mathematical probabilistic dependency between random variables. This

view of the knowledge base as relative, instead of absolute, provides the authority to manipulate probabilities as weights in order to ensure the accuracy of the solutions.[8]

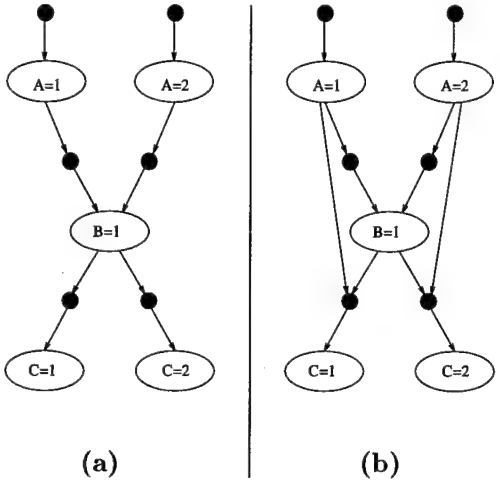


Figure 4.2 **Thrashing.** Here the BKB in part (a) will thrash when conflicting test cases are presented. Part (b) shows one possible remedy to the thrashing.

4.4.3 Thrashing. As mentioned previously, due to the incompleteness allowed in BKBs, thrashing can occur during the validation. When thrashing occurs, the knowledge base oscillates (thrashes) between two or more states, where each state is lacking some of the qualities (accuracies) of at least one other state. For an example, see Figure 4.2. In this example thrashing occurs in the BKB represented by part (a) when the following two test cases are included in the requirements:

1. Evidence: $A = 1$—Expected Answer: $C = 1$
2. Evidence: $A = 2$—Expected Answer: $C = 2$

Validation of the first test case ensures that the following inequality is true:

$$P(C = 1 \mid B = 1) > P(C = 2 \mid B = 1).$$

[8] It should also be noted that the knowledge acquisition process does not acquire precise numeric probabilities, but uses linguistic parameters to record an approximation of the value [46, 2]. During verification, these probabilities are changed to ensure consistency. Since no exact probabilities are ever see by the user, MACK's and BVAL's manipulations are invisible.

This must be the case since only one instance of C can be included in any solution. Conversely, validation of the second test case ensures:

$$P(C = 1 \mid B = 1) \; < \; P(C = 2 \mid B = 1).$$

Left unchecked, this thrashing would continue indefinitely. A skillful user may detect this condition and choose to remedy the situation, or pick the best of the thrashing states. However, automatic detection of this condition is possible by monitoring the changes occurring in the knowledge base each time the test suite is validated. Once the test cases causing thrashing are identified, an analysis of the solutions generated by the test cases can provide pointers to the areas responsible for the thrashing. Figure 4.2(b) shows one possible remedy to this particular situation. Future work in this area will attempt to isolate and define conditions leading to thrashing, with possible automated or semi-automated correction.

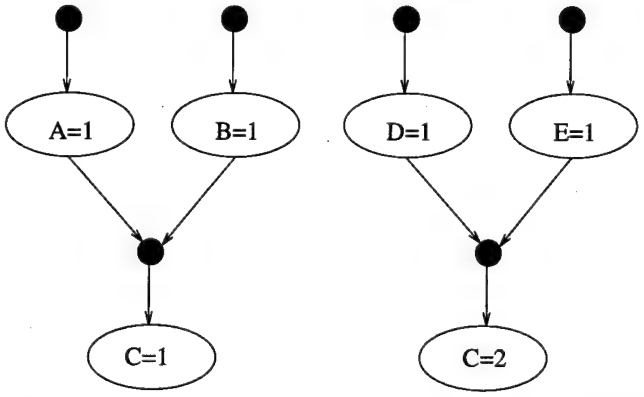


Figure 4.3 **Potential Thrashing.** This BKB is susceptible to thrashing if a test case like: Evidence: $A = 1$, $B = 1$; Expected Answer: $C = 2$ is included in its requirements

While the aforementioned example showed a thrashing situation detectable during validation, it is possible to detect some potentially thrash-causing test cases before any validation is done.

The BKB shown in Figure 4.3 is susceptible to thrashing if the following test case is present in its requirements:

1. Evidence: $A = 1$, $B = 1$—Expected Answer: $C = 2$

Validation of this test case alone would not result in thrashing; however, if a test case such as

2. Evidence: $D = 1$, $E = 1$—Expected Answer: $C = 1$

were added to the requirements at a later time, thrashing would result. BVAL checks for this potentially thrash-causing relationship between the evidence and the expected answer, warning the user appropriately (see Section C.2).

4.5 Incompleteness

BVAL recognizes incompleteness in a BKB by the presence of assumed RV-states in the solution. As mentioned earlier (Section 3.5.3), RV-states are only assumed when no other solution can be made without an assumption. For an RV-state to be assumed, the conditions required to activate it would have been prohibited by other assignments in the knowledge base. (Recall that a solution is a global optimization; no assumptions will be made unless absolutely necessary.) Since BVAL's methodology ensures incompleteness will not be addressed unless the solution is correct, the path created by the correct solution contains an assumption. BVAL directs the user to this assumption(s) and provides a means for investigating the surrounding knowledge.

The BKB in Figure 4.4(a) generates an assumption when the following test case is attempted to be validated:

1. Evidence: $E = 1$—Expected Answer: $B = 1$.

When BVAL detects this situation, the user is prompted with a menu of options to take:

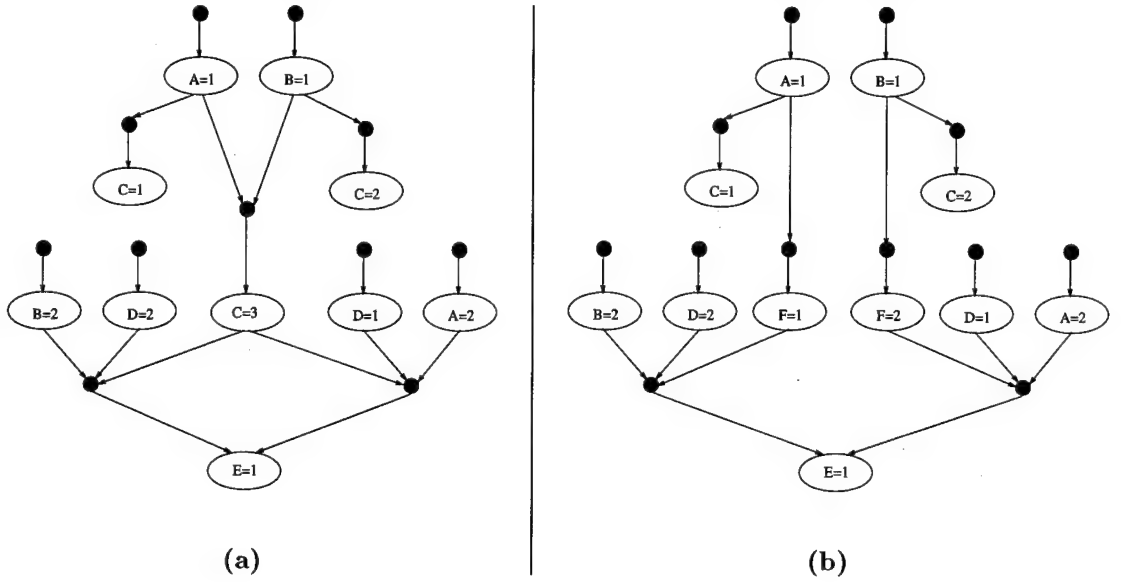


Figure 4.4 **Incomplete Knowledge (1).** In part (a), the test case of: Evidence: $E = 1$; Expected Answer: $B = 1$, results in $E = 1$ being assumed. Part (b) shows one possible remedy to this situation chosen after interaction with BVAL. (Specifically Incomplete menu choice 2.)

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

This menu offers the user three different ways of examining the assumptions that are impeding the validation of a particular test case. The first option lists all the assumptions present in the known correct solution. The second option details the conditions needed to avoid the assumption, and what conditions were actually present in the solution. The final inspection option lists the RV-level connections of any other RV-states belonging to the same RV as the assumption. For this test case and knowledge base, each option appears as:

Choice: 1

The assumed components and their assumed states:

E : 1

Choice: 2

Select assumed component and value:

1 – E : 1

Choice: 1

For 'E : 1', one of the following support conditions must be true:

Support condition 1 needs these true:

A : 2
C : 3
D : 1

Support condition 2 needs these true:

B : 2
C : 3
D : 2

However, in the solution, the following were true:

B : 1
C : 2

And the following were not in the solution:

A : 2
D : 1
D : 2

Choice: 3

Select assumed component and value:

1 – E : 1

Choice: 1

E : 1 is connected to the following components:

A
B
C
D

In this particular case, option two reveals that both support conditions depend on $C = 3$; however, $C = 2$ is in the solution. One possible remedy to this situation is to remove $C = 3$ from the knowledge base and replace it with two separate RV-states: $F = 1$ and $F = 2$. To carry out this task, the user can choose option nine, which provides the familiar MACK tools to edit the knowledge base.

Choice: 9

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 1

Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 3

Please pick an instantiation of C:

1 – C = 1
2 – C = 2
3 – C = 3
0 – Abort

Choice: 3

Deleting instantiation of C = 3

In addition to removing RV-state $C = 3$, RV F and its instances must be added to the BKB along with editing the support conditions for $E = 1$. After accomplishing this task, the BKB is transformed into Figure 4.4(b). (For a complete transcript of this session see Section C.3.1.)

Once the knowledge base editing is complete, option zero allows the validation process to continue. This option initiates the iterative re-validation mentioned in Section 4.2. In this scenario, the unacceptable incompleteness is successfully removed, and validation can continue.

Choice: 0

Querying BKB...
Correct Answer and solution is Computable.
Validated!

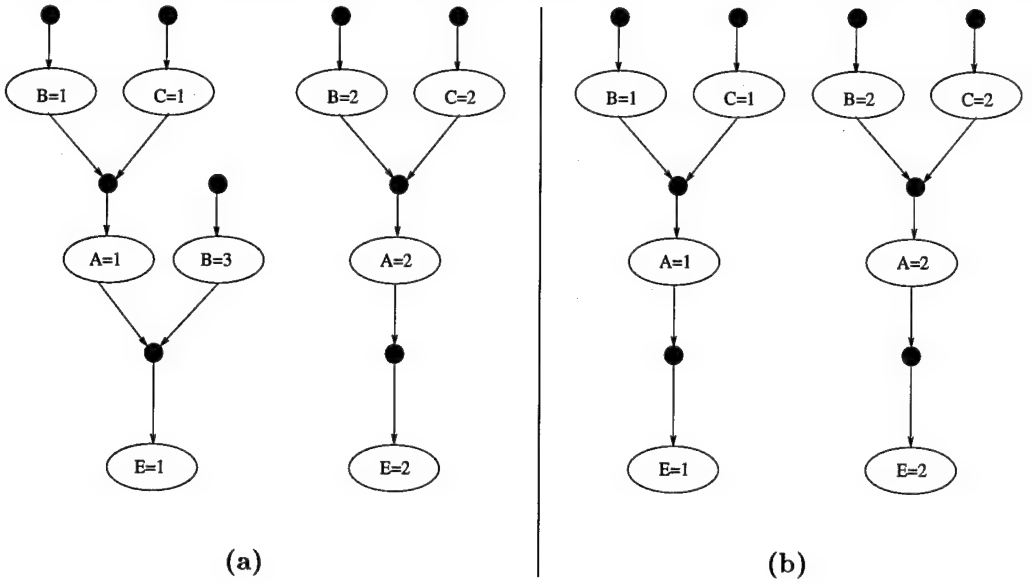


Figure 4.5 **Incomplete Knowledge (2).** Here the BKB in part (a) demonstrates incompleteness when presented with the test case: Evidence: $E = 1$; Expected Answer: $B = 1$, $C = 1$. Part (b) shows one possible solution prompted by interaction with BVAL. (Specifically Incomplete menu choice 3.)

The BKB in Figure 4.5(a) also suffers from incompleteness when presented with:

1. Evidence: $E = 1$—Expected Answer: $B = 1, C = 1$.

In this example, the incompleteness menu choice three reveals information critical to the correction of the problem.

Choice: 3

Select assumed component and value:

1 – E : 1

Choice: 1

E : 1 is connected to the following components:

A

B

E : 2 is connected to the following components:

A

Here the listing of other connections in the same RV show the assumed RV-state $E = 1$ is dependent upon RVs A and B, while $E = 2$ is only dependent upon A. It is possible that the dependency relationship between B and E was constructed in error—exactly the type of situation that this menu choice revealed. Correction of this problem proceeds similarly to the earlier example, with the BKBs final configuration shown in Figure 4.5(b). (This entire transcript is in Section C.3.2.)

Although incompleteness is allowed in the knowledge base, it is only permitted in as much as it does not interfere with the requirements of the knowledge base. In other words, the assumed RV-states indicate incompleteness that must be eliminated in order to meet the requirements. Eliminating the assumptions requires interaction with the user, and as such, this task is the most visible of BVAL's operations. While the pointers to areas of incomplete knowledge are helpful, the text only output is a limitation. Future work will address a more graphical interface, and additional mechanisms for spotlighting incompleteness.

V. Conclusions

This research develops a methodology and an automated tool for the validation of probabilistic knowledge bases. This tool, BVAL, incorporates aspects of both rule-based and neural network validation techniques to provide automatic correction and localization of problematic knowledge. The validation is accomplished by a formal comparison of the knowledge base against its requirements; where the requirements are composed of test cases.

The requirements (test suite) are processed one test case at a time, ensuring the knowledge base is as complete and as correct as needed to validate each requirement. When a requirement highlights areas of incompleteness, BVAL intelligently interacts with the user, assisting in the correction of the problem. If BVAL locates a requirement that can be satisfied by an adjustment of the probabilities, the adjustment is made automatically. This adjustment accounts for the strict probabilistic nature of the knowledge base and the finite resolution of any machine without bothering the user. At completion of validation, BVAL guarantees the knowledge base has remained consistent and is now 100% accurate with respect to the requirements.

BVAL is designed to integrate with the previous tools developed for Bayesian Knowledge Bases [46, 2], and to operate in an iterative manner. This incremental validation allows the user to gain confidence in the knowledge base as more and more knowledge is incorporated. After the knowledge base enters the maintenance portion of its lifecycle, BVAL remains useful by validating changes and additions to the knowledge base and/or its requirements.

The flexibility of Bayesian Knowledge Bases incorporated with the usefulness of BVAL over the knowledge base lifecycle, charts a course for the development and maintenance of probabilistic knowledge bases well into the future.

VI. Recommendations

6.1 New Type of Knowledge Acquisition

The current validation scheme which combines traditional test-case validation with reinforcement learning opens many doors of opportunity for future work. Since the probabilities are set initially with only linguistic parameters, and modified for consistency and validation, there seems to be a possibility for a new approach to knowledge acquisition for probabilistic belief networks. This new approach would need only to interact with the expert or a database of information to get the "symbolic knowledge", i.e., the random variables, their states, and the dependencies between them [5, 27, 1, 21]. The probabilities could all be set by the validation portion of the knowledge acquisition cycle. This scheme seems to solidify a relationship between belief networks and neural networks. The belief network keeps its knowledge in weighted connections and symbols, whereas the neural network keeps all of its knowledge in weighted connections. Use of this new knowledge acquisition technique would seem, in large part, to avoid one of the more troubling issues in probabilistic knowledge acquisition.

6.2 Reduction Factor Calculations

While the current method of calculating the reduction factor works properly, it can reduce more than necessary. This is due to the fact that any solution to the given evidence (possibly none) can have a varying number of RV-states. This varying number could be as small as one, or as large as the longest path(s) through all the evidence. Because of this possibility, the reduction factor currently used is the entire difference (as a percentage) between the correct and incorrect probabilities. However, for example, if the two solutions in question are the first and second most probable, and the incorrect solution (most probable) contains five RV-states, only the fifth-root of this percentage is needed to achieve the desired results, since the reduction is multiplied five times. Conversely, if the correct and incorrect solutions are third and first most probable respectively, and

the second most probable solution contains three RV-states, then the fifth-root is not guaranteed to affect the desired change. In this case the third-root will definitely work, but may itself be larger than needed. Further work in this area would define a calculation that still guarantees reinforcement in one pass and changes the probabilities a minimal amount.

6.3 Assisting the User with Incompleteness

Solving incompleteness issues, assumptions and thrashing, can actually lead to more incompleteness. This is due to the fact that knowledge is not monotonic; i.e. more knowledge does not guarantee more completeness. A great improvement to BVAL's current methodology, would be a knowledge contradiction detector. This detector would examine changes made to the knowledge base in an attempt to determine if additional incompleteness was created by "fixing" a different problem. This contradiction detector would provide the user with another tool for locating and remedying incompleteness. Any additional tools or assistance are of great benefit when requiring user interaction.

Another aid to remedying incompleteness would be a knowledge viewer. This viewer would graphically show the knowledge to the user, providing more precise insight into the actual structure of the knowledge base. Being able to see a portion of the knowledge base where assumptions or thrashing are occurring can provide a great deal more information than simple text-based methods. Having as many tools as possible at the disposal of the user can do nothing but help the mandatory interactions.

6.4 Extending the Validation Methodology

Aside from techniques in aiding the resolution of incompleteness problems, the validation method itself can be extended. One possible extension is to enable validation of more than just the first (most probable) solution. This extension would operate using a test case that contains

evidence and expected answer, as before, with additional constraints on the "importance" of the expected answer. This "importance" information could specify the expected answer "must be the most probable solution to the given evidence", or "must be in the top three most probable solutions", or "must not be in the top two most probable solutions", etc.. These constraints could allow for a more complete validation of the knowledge base, and could provide a means for multiple experts to agree on its accuracy.

A more pragmatic continuation of this validation work would be the investigation of "relevant validation". While the current scheme can be applied iteratively through the knowledge acquisition process, in order to ensure the best accuracy all test cases must be validated concurrently. A better approach would assure the accuracy of the knowledge base without having to use every test case. This technique could maintain all test cases ever presented to the system, and re-validate only those (relevant) cases effected by a change to the knowledge base. This approach could reduce the amount of time and effort spent in the validation phase, allowing the system to be maintained and used more efficiently.

6.5 Widening Applications

BVAL's methodology was developed for a probabilistic knowledge representation, Bayesian Knowledge Bases, and it easily extends to to traditional Bayesian Networks. In fact, since Bayesian Networks do not incorporate incompleteness, BVAL's functionality would actually be reduced to accommodate a more restrictive knowledge representation. BVAL's approach to reinforcement learning is applicable to Bayesian Networks, and any thrashing will still indicate gaps in the knowledge base. Since thrashing is the only catalyst for user interaction, BVAL running on a Bayesian Network would potentially be fully automated. This type of automated learning/validation could be of great usefulness to the probabilistic knowledge base community [25, 24].

6.6 Look-Beyond Database

As mentioned previously (Section 3.5.3), the Belief Extension inferencing method makes use of a look-beyond factor. This factor assists the inference engine by directing the inclusion of relevant RV-states in the solution based on the evidence. While the user is free to change the look-beyond value for any particular query, the test suite can be used to establish a database of initial look-beyond values. This database would be composed of the evidence and look-beyond value of each test case.[1] Once this database is created, the inferencing method (or a separate preprocessor) can apply some type of matching constraints to arrive at an estimate of the look beyond factor for any query—even one not in the test suite. This additional capability would give the overall system the ability to approximate the relevant components of a solution in real time.

[1] The look-beyond value is calculated by a simple algorithm that observes the structure of the BKB in relation to the evidence and expected answer.

Bibliography

1. Aliferis, Constantin F. and Gregory F. Cooper. "An evaluation of an algorithm for inductive learning of Bayesian belief networks using simulated data sets." *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*. 8–14. 1994.
2. Banks, Darwyn O. *Acquiring Consistent Knowledge for Bayesian Forests*. MS thesis, AFIT/GSO/ENG/95M-01. Graduate school of engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, March 1995.
3. Barr, Avron, et al., editors. *The Handbook of Artificial Intelligence, 4*. Addison-Wesley Publishing Company, Inc., 1989.
4. Botten, Nancy, et al. "Knowledge bases: Integration, verification, and partitioning," *European Journal of Operational Research, 42*:111–128 (1989). ([20:pages 150–158]).
5. Bouckaert, Remco R. "Properties of Bayesian Belief Network Learning Algorithms." *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*. 102–109. 1994.
6. Buchanan, Bruce G. and Edward H. Shortliffe, editors. *Rule-Based Expert Systems*. Addison-Wesley Publishing Company, 1984.
7. Burnell, Lisa and Eric Horvitz. "Structure and Chance: Melding Logic and Probability for Software Debugging," *Communications of the ACM, 38*(3):31–41, 57 (March 1995).
8. Charniak, Eugene. "Bayesian Networks without Tears," *AI Magazine*, 50–63 (Winter 1991).
9. Charniak, Eugene and Drew McDermott. *Introduction to Artificial Intelligence*. Reading, MA: Addison-Wesley, 1984.
10. Cooper, Gregory F. "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks," *Artificial Intelligence, 42*:393–405 (1990).
11. Davis, Randall. "Interactive transfer of expertise: Acquisition of new inference rules.," *Artificial Intelligence, 12*:121–158 (1979).
12. Dempster, A. P. "Upper and Lower Probabilities Induced by Multivalued Mappings," *Annals of Mathematical Statistics, 38*:325–329 (1967).
13. Duda, R., et al. "Model Design in the PROSPECTOR Consultant System for Mineral Exploration." *Expert Systems in the Micro-Electronic Age* edited by D. Michie, 153–167, Edinburgh University Press, 1979.
14. Fielding, Ken. "Hold-Out or Hold-One-Out?." Graduate School of Engineering, Air Force Institute of Technology, April 1993.
15. Fung, Robert and Brendan DelFavero. "Applying Bayesian Networks to Information Retrieval," *Communications of the ACM, 38*(3):42–48, 57 (March 1995).
16. Giarratano, Joseph and Gary Riley. *Expert Systems: Principles and Programming* (Second Edition). PWS Publishing Company, Boston, MA, 1994.
17. Gonzalez, Avelino J. and Douglas D. Dankel. *The Engineering of Knowledge-Based Systems*. Prentice Hall, 1993.
18. Green, Christopher J. R. "On the Use of Requirements in Development of Knowledge-Based Systems." *Proceedings of the AAAI Workshop on Validation and Verification of Expert Systems*. 1988. ([20:pages 240–248]).
19. Guida, Giovanni and Ginacarlo Mauri. "Evaluating Performance and Quality of Knowledge-Based Systems: Foundation and Methodology," *IEEE Transactions on Knowledge and Data Engineering, 5*(2):204–224 (April 1993).

20. Gupta, Uma G., editor. *Validating and Verifying Knowledge-Based Systems*. IEEE Computer Society Press, 1991. This compilation has many sources included in their entirity: [36, 4, 18].

21. Haddawy, Peter. "Generating Bayesian Networks from Probability Logic Knowledge Bases." *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*. 262–269. 1994.

22. Healey, Patrice M. and Edwin J. Jacobson. *Common Medical Diagnoses: An Algorithmic Approach*. W. B. Saunders Company, 1990.

23. Hecht-Nielsen, Robert. *Neurocomputing*. Addison-Wesley Publishing Company, 1991.

24. Heckerman, David. "A Bayesian Approach to Learning Causal Networks." *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference*. 285–295. 1995.

25. Heckerman, David. *A Tutorial on Learning Bayesian Networks*. Technical Report MSR-TR-95-06, Microsoft Research, October 1995. (Revised Version).

26. Heckerman, David, et al. "Decision-Theoretic Troubleshooting," *Communications of the ACM*, *38*(3):49–57 (March 1995).

27. Heckerman, David, et al. "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data." *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*. 293–301. 1994.

28. Kinderknecht, Stanley D. *Semantic Interpretation of an Artificial Neural Network*. MS thesis, AFIT/GCS/ENG/95D-07. Graduate school of engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1995.

29. Lachenbruch, Peter A. "An almost unbiased method of obtaining confidence intervals for the probability of misclassification in discriminant analysis," *Biometrics*, *23*:639–645 (December 1967).

30. Lindsay, Robert K., et al. "DENDRAL: a case study of the first expert system for scientific hypothesis formation," *Artificial Intelligence*, *61*:209–261 (1993).

31. McDermott, John and Judith Bachant. "R1 Revisited: Four Years in the Trenches," *AI Magazine*, *5*(3):21–32 (Fall 1984).

32. Merlevede, P. and J. Vanthienen. "A Structured Approach to Formalization and Validation of Knowledge." *Developing and Managing Expert System Programs*. 149–158. Los Alamitos, CA: IEEE Computer Society Press, 1991.

33. Nazareth, Derek L. "Issues in the verification of knowledge in rule-based systems," *International Journal of Man-Machine Studies*, *30*:255–271 (1989).

34. Nazareth, Derek L. "Investigating the Applicability of Petri Nets for Rule-Based System Verification," *IEEE Transactions on Knowledge and Data Engineering*, *4*(3):402–415 (June 1993).

35. Ng, Keung-Chi and Bruce Abramson. "Uncertainty Management in Expert Systems," *IEEE Expert*, *5*(2):29–48 (April 1990).

36. O'Keefe, Robert M., et al. "Validating Expert System Performance," *IEEE Expert*, 81–89 (Winter 1987). ([20:pages 2–11]).

37. Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1991. (Revised Second Printing).

38. Ramoni, Marco. "Ignorant Influence Diagrams." *IJCAI-95: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. 1869–1875. 1995.

39. Ramoni, Marco and Alberto Riva. "Belif Maintenance in Bayesian Networks." *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. 498–505. 1994.

40. Rogers, Steven K. Conversations with author, Fall 1995.

41. Rogers, Steven K. and Matthew Kabrisky. *An Introduction to Biological and Artificial Neural Networks for Pattern Recognition*. SPIE Optical Engineering Press, 1991.

42. Rumelhart, David E., et al. *Parallel Distrubuted Processing, 1*. MIT Press, 1986.

43. Santos, Jr., Eugene. "A Fully Integrated Probabilistic Framework for Expert Systems Development." Research proposal from Air Force Institute of Technology to Air Force Office of Scientific Research. March 1993.

44. Santos, Jr., Eugene. *Computing with Bayesian Multi-Networks*. Technical Report AFIT/EN/TR93-10, Department of Electrical and Computer Engineering, Air Force Institute of Technology, November 1993.

45. Santos, Jr., Eugene. "Representing and Reasoning with Bayesian Knowledge Bases." Submitted for Publication, 1995.

46. Santos, Jr., Eugene and Darwyn O. Banks. "Acquiring Consistent Knowledge in the Face of Uncertainty," *IEEE Transactions on Knowledge and Data Engineering* (1995). (Submitted to).

47. Santos, Jr., Eugene, et al. *On a Distributed Anytime Architecture for Probabilistic Reasoning*. Technical Report AFIT/EN/TR95-02, Department of Electrical and Computer Engineering, Air Force Institue of Technology, 1995.

48. Shafer, Glenn. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

49. Sy, Bon K. "Reasoning MPE to Multiply Connected Belief Networks Using Message Passing." *Proceedings of the Tenth National Conference on Artificial Intelligence*. 1992.

50. Yager, Ronald R. and Henry L. Larsen. "On Discovering Potential Inconsistencies in Validating Uncertain Knowledge Bases by Reflecting on the Input," *IEEE Transactions on Systems, Man, and Cybernetics*, *21*(4):790–801 (1991).

51. Zadeh, Lotfi A. "Fuzzy Sets," *Information and Control*, *8*(3):338–353 (1965).

52. Zlatareva, Neli. "An Effective Logical Framework for Knowledge-Based Systems Verification," *International Journal of Expert Systems*, 7(3):239–260 (1994).

Appendix A. PESKI

The Bayesian Knowledge Base has been put into a new expert system architecture called PESKI (Probabilities, Expert System, Knowledge and Inference) and is currently being explored as a viable knowledge representation[43].

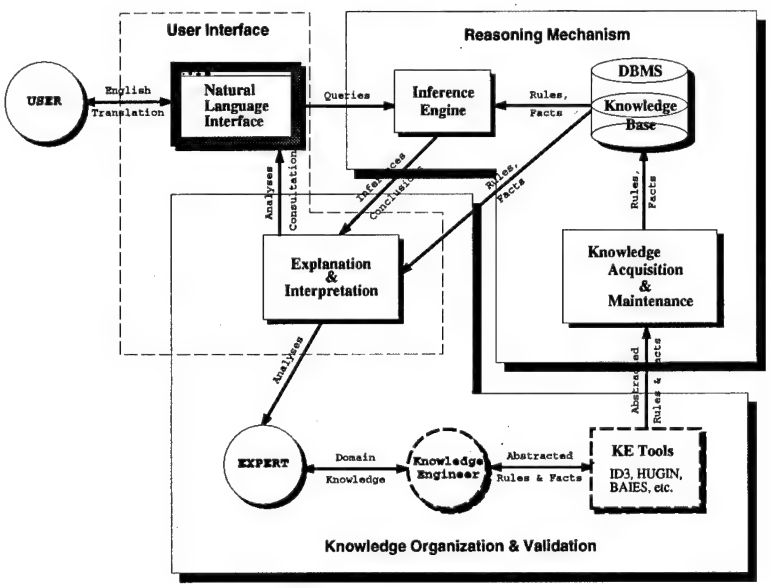


Figure A.1 **The PESKI architecture.** The broken boarder components are considered optional.

As shown in Figure A.1), the PESKI architecture is composed of three major sub-systems and four main components. The main components perform their obvious roles:

- **Natural Language Interface**—provides for communication between the user and the system by translating English questions into inference queries; inference results into English responses.
- **Inference Engine**—performs the "reasoning" associated with answering queries; controls the selection of strategies for reasoning.

- **Explanation & Interpretation**—tracks the reasoning paths utilized by the inference engine, and can compare them against the knowledge base as needed.
- **Knowledge Acquisition & Maintenance**—provides the capability to automatically incorporate new or updated knowledge into the knowledge base.

The components are grouped into overlapping subsystems since there seems to be a natural intersection between them:

- *User Interface*—composed of the Natural Language Interface and the Explanation & Interpretation components, this subsystem will provide the primary conduit for interaction with the user.
- *Knowledge Organization & Validation*—composed of the Explanation & Interpretation component along with the human expert, optional knowledge engineer and knowledge engineering tools. Organization is accomplished via interaction with the Knowledge Acquisition and Maintenance component. Validation (BVAL) is similarly accomplished by also including feedback from the Reasoning Mechanism through Explanation & Interpretation for requirements matching.
- *Reasoning Mechanism*—composed of the Inference Engine and the Knowledge Acquisition & Maintenance components. Including the Knowledge Acquisition & Maintenance component in the subsystem provides a large degree of information hiding with respect to the specific knowledge representation being used as the knowledge base. Additionally, our current and previous work suggests that some form of reasoning, possibly learning, may need to take place during the acquisition and maintenance of knowledge. Furthermore, since consistency is reconciled during acquisition, and incompleteness is remedied by interaction with the Knowledge Acquisition & Maintenance component, the arrangement in this subsystem seems valid.

Easily seen is this architecture's flexibility compared to previous designs. PESKI has proven to be sufficient for constructing knowledge-based systems in nearly any domain.

Appendix B. Connected Region

The connected region of a Bayesian Knowledge Base (BKB) can be informally thought of as the portion of the knowledge base that could have any bearing on the solution given the evidence. This can easily be visualized by considering a knowledge base containing medical and automotive knowledge. It seems obvious that the medical portion of the knowledge base would be wholly disconnected from the automotive portion. For any validation done on this knowledge base, individual test cases would probably only need to be concerned with either the medical or automotive portion. Therefore, any reinforcement, up-scaling, or Belief Extension inferencing would only ever be concerned with one portion of the knowledge base—the connected region.

While this informal discussion provides an intuitive feel for the connected region, it lacks precision. It is possible to formally define the connected region of a Bayesian Knowledge Base. Observing only the structure of the knowledge base,[1] a BKB can be defined as an ordered triple

$$B = (I, S, D)$$

where I is the set of RV-states, instance-nodes (I-nodes); S is the set of probabilistic dependency nodes, support-nodes (S-nodes); and D is a set of ordered pairs describing the dependencies between I-nodes and S-nodes. The set of I-nodes is described as

$$I = \{I_{1,1}, I_{1,2}, \ldots, I_{1,N_1}, I_{2,1}, I_{2,2}, \ldots, I_{2,N_2}, \ldots, I_{M,1}, I_{M,2}, \ldots, I_{M,N_M}\}$$

where M is the number of RVs and N_i is the number of states of RV I_i. The set of S-nodes is described as

$$S = \{S_{(I_{1,1})_1}, S_{(I_{1,1})_2}, \ldots, S_{(I_{1,1})_{P_{1,1}}}, S_{(I_{1,2})_1}, \ldots, S_{(I_{1,2})_{P_{1,2}}}, \ldots, S_{(I_{M,1})_1}, \ldots, S_{(I_{M,N_M})_1}, \ldots, S_{(I_{M,N_M})_{P_{M,N_M}}}\}$$

[1] The formalisms for ensuring consistency and proper probabilistic representation are found in [44, 46, 2].

where $P_{i,j}$ is the number of probabilities of the form $\mathrm{P}(I_{i,j}|\Gamma)$, $\Gamma \in I$ in the knowledge base. Lastly, the set of dependencies is described as

$$D = \{(I_\alpha, S_{\gamma_\beta}), (S_{\gamma_\beta}, I_\gamma) \mid S_{\gamma_\beta} = \mathrm{P}(I_\gamma|I_\alpha, \Gamma)\} \cup \{(S_{\gamma_\beta}, I_\gamma) \mid S_{\gamma_\beta} = \mathrm{P}(I_\gamma)\}$$

where α and $\gamma = (i,j)$; $i = 1 \ldots M$, $j = 1 \ldots N_i$ with $\alpha \neq \gamma$, and $\beta = 1 \ldots P_{i,N_i}$.

To illustrate, the BKB in Figure B.1 is described as:

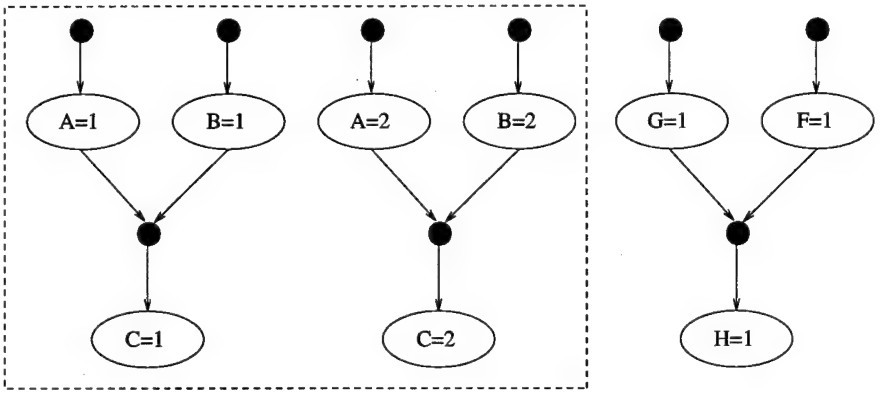


Figure B.1 **Bayesian Knowledge Base.** An illustrative BKB with 6 RVs, 9 RV-states, and 9 probabilities (3 of which are dependent probabilities). The connected region for evidence of $E = \{A_1, B_1\}$ is outlined.

$$\begin{aligned}
B &= (I, S, D) \quad \text{where} \\
I &= \{A_1, A_2, B_1, B_2, C_1, C_2, F_1, G_1, H_1\}^2 \\
S &= \{\mathrm{P}(C_1|A_1, B_1), \mathrm{P}(C_2|A_2, B_2), \mathrm{P}(H_1|F_1, G_1), \mathrm{P}(A_1), \mathrm{P}(A_2), \\
&\quad \mathrm{P}(B_1), \mathrm{P}(B_2), \mathrm{P}(F_1), \mathrm{P}(G_1)\} \\
&\quad = \{S_{C_1}, S_{C_2}, S_{H_1}, S_{A_1}, S_{A_2}, S_{B_1}, S_{B_2}, S_{F_1}, S_{G_1}\} \\
D &= \{(A_1, S_{C_1}), (B_1, S_{C_1}), (S_{C_1}, C_1), (A_2, S_{C_2}), (B_2, S_{C_2}), (S_{C_2}, C_2), \\
&\quad (E_1, S_{E_1}), (F_1, S_{E_1}), (S_{E_1}, G_1), (S_{A_1}, A_1), (S_{A_2}, A_2), (S_{B_1}, B_1), \\
&\quad (S_{B_2}, B_2), (S_{F_1}, F_1), (S_{G_1}, G_1)\}.
\end{aligned}$$

[2]Here the $I_{i,j}$ is replaced with α_j where α corresponds to the RV I_i.

The connected region is the partial BKB

$$B' = (I', S', D')$$

formed by an analysis of the evidence submitted in a query

$$E = \{E_1, E_2, \ldots, E_N\} \quad \text{where} \quad E_i \in I \quad \text{forall} \quad i.$$

The connected region is defined as:

$$\begin{array}{rcl}
B' & = & (I', S', D') \quad \text{where} \\
I' & = & E \cup \{I_{i_j} \mid \forall I_{i_j} \in I' :: [(I_{i_j}, S_k) \in S \ \wedge \ (S_k, I_{x_y}) \in S] \rightarrow I_{i_j} \in I'\} \ \cup \\
 & & \{I_{i_j} \mid \forall I_{i_j} \in I' :: [(I_{x_y}, S_k) \in S \ \wedge \ (S_k, I_{i_j}) \in S] \rightarrow I_{i_j} \in I'\} \ \cup \\
 & & \{I_{i_j} \mid \forall I_{i_j} \in I';\ \forall k \in 1..N_i :: I_{i_j} \in I' \rightarrow I_{i_k} \in I'\} \\
S' & = & \{S_k \mid \forall I_{i_j} \in I' :: (I_{i_j}, S_k) \in S \rightarrow S_k \in S'\} \ \cup \\
 & & \{S_k \mid \forall I_{i_j} \in I' :: (S_k, I_{i_j}) \in S \rightarrow S_k \in S'\} \\
D' & = & \{(I_x, S_y) \mid \forall I_x \in I';\ \forall S_y \in S' :: (I_x, S_y) \in D \rightarrow (I_x, S_y) \in D'\} \ \cup \\
 & & \{(S_y, I_x) \mid \forall I_x \in I';\ \forall S_y \in S' :: (S_y, I_x) \in D \rightarrow (S_y, I_x) \in D'\}
\end{array}$$

In simpler terms, the connected region is all RV-states of every RV connected to the evidence, plus all of their connections, etc.. For example, if the evidence for a particular query is the set

$$E = \{A_1, B_1\}$$

then the connected region of the BKB from Figure B.1 is:

$$\begin{array}{rcl}
B' & = & (I', S', D') \quad \text{where} \\
I' & = & \{A_1, A_2, B_1, B_2, C_1, C_2\} \\
S' & = & \{\mathrm{P}(C_1|A_1, B_1), \mathrm{P}(C_2|A_2, B_2), \mathrm{P}(A_1), \mathrm{P}(A_2), \\
 & & \mathrm{P}(B_1), \mathrm{P}(B_2)\} \\
 & & = \{S_{C_1}, S_{C_2}, S_{A_1}, S_{A_2}, S_{B_1}, S_{B_2}\} \\
D' & = & \{(A_1, S_{C_1}), (B_1, S_{C_1}), (S_{C_1}, C_1), (A_2, S_{C_2}), (B_2, S_{C_2}), (S_{C_2}, C_2), \\
 & & (S_{A_1}, A_1), (S_{A_2}, A_2), (S_{B_1}, B_1), (S_{B_2}, B_2)\}.
\end{array}$$

A similar notion can be easily developed for traditional Bayesian Networks.

Appendix C. BVAL Sessions

C.1 Reinforcement Session

Please enter the EVIDENCE
Please pick a component:

1 – Acute Bronchitis
2 – Chronic Bronchitis
3 – Cough
4 – CXR
5 – History and Physical
6 – Interstitial Fibrosis
7 – Malignant Disease
8 – Obstructive Pulmonary Disease
9 – PFTs
10 – Pneumonia
11 – Sputum or Tissue Culture
0 – Abort

Choice: 3

Please pick an instantiation of Cough:

1 – Cough = TRUE
0 – Abort

Choice: 1

Please enter the EVIDENCE
Please pick a component:

1 – Acute Bronchitis
2 – Chronic Bronchitis
3 – Cough
4 – CXR
5 – History and Physical
6 – Interstitial Fibrosis
7 – Malignant Disease
8 – Obstructive Pulmonary Disease
9 – PFTs
10 – Pneumonia
11 – Sputum or Tissue Culture
0 – Abort

Choice: 0

...calculating connected region...

Please enter the expected ANSWER
Please pick a component:

1 – Acute Bronchitis
2 – Chronic Bronchitis

3 – Cough
4 – CXR
5 – History and Physical
6 – Interstitial Fibrosis
7 – Malignant Disease
8 – Obstructive Pulmonary Disease
9 – PFTs
10 – Pneumonia
11 – Sputum or Tissue Culture
0 – Abort

Choice: 2

Please pick an instantiation of Chronic Bronchitis:

1 – Chronic Bronchitis = TRUE
0 – Abort

Choice: 1

Please enter the expected ANSWER
Please pick a component:

1 – Acute Bronchitis
2 – Chronic Bronchitis
3 – Cough
4 – CXR
5 – History and Physical
6 – Interstitial Fibrosis
7 – Malignant Disease
8 – Obstructive Pulmonary Disease
9 – PFTs
10 – Pneumonia
11 – Sputum or Tissue Culture
0 – Abort

Choice: 0

Querying BKB...
Incorrect Answer and solution is Computable.

Querying BKB...
Solution known to be correct and it is Computable.

Applying Reinforcement Learning...
Reduction factor is 0.234005

Applying an up-scale factor of 1.07892

Querying BKB...
Correct Answer and solution is Computable.
Validated!

C.2 Potential Thrashing Condition

Please enter the EVIDENCE
Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 1

Please pick an instantiation of A:

1 – A = 1
0 – Abort

Choice: 1

Please enter the EVIDENCE
Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 2

Please pick an instantiation of B:

1 – B = 1
0 – Abort

Choice: 1

Please enter the EVIDENCE
Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 0

...calculating connected region...

Please enter the expected ANSWER
Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 3

Please pick an instantiation of C:

1 – C = 1
2 – C = 2
0 – Abort

Choice: 2

********** W A R N I N G **********

The inclusion of C : 2
in the expected answer makes this test case
potentially thrash-causing if a conflicting
test case is included in the requirements.
****

Please enter the expected ANSWER
Please pick a component:

1 – A
2 – B
3 – C
4 – E
5 – F
0 – Abort

Choice: 0

C.3 Incompleteness

C.3.1 From Figure 4.4.

Please enter the EVIDENCE
Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 5

Please pick an instantiation of E:

1 – E = 1
0 – Abort
Choice: 1

Please enter the EVIDENCE
Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 0

...calculating connected region...

Please enter the expected ANSWER
Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 2

Please pick an instantiation of B:

1 – B = 1
2 – B = 2
0 – Abort
Choice: 1

Please enter the expected ANSWER
Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 0

Querying BKB...
Incorrect Answer and solution is NOT Computable.

Querying BKB...
Solution known to be correct and it is NOT Computable.

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 1

The assumed components and their assumed states:

E : 1

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 2

Select assumed component and value:

1 – E : 1

Choice: 1

For 'E : 1', one of the following support conditions must be true:

Support condition 1 needs these true:

A : 2
C : 3
D : 1

Support condition 2 needs these true:

B : 2
C : 3
D : 2

However, in the solution, the following were true:

B : 1
C : 2

And the following were not in the solution:

A : 2
D : 1
D : 2

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 3

Select assumed component and value:

1 – E : 1

Choice: 1

E : 1 is connected to the following components:

A
B
C
D

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 9

****
If any changes are made to the knowledge base*
it must be consistent before continuing. *
****

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 1

Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Abort

Choice: 3

Please pick an instantiation of C:

1 – C = 1
2 – C = 2
3 – C = 3
0 – Abort

Choice: 3

Deleting instantiation of C = 3

It is possible that the E is 1 depending upon . . .

A = 2
C = 3
D = 1

Do you still wish to delete 3 as valid value for C? Y/N y

Invalidate this entire support condition for E being 1? [Enter 0]
or
Simply remove C being 3 from the above list of support conditions? [Enter 1]

Choice: 1

Please complete the sentence below from the following list of choices:

0 – inconceivable
1 – not likely
2 – possible
3 – probable
4 – almost certain

It is _____ that the E is 1 depending upon . . .

A = 2
D = 1

Choice: 2

It is possible that the E is 1 depending upon . . .

B = 2
C = 3
D = 2

Do you still wish to delete 3 as valid value for C? Y/N y

Invalidate this entire support condition for E being 1? [Enter 0]
or
Simply remove C being 3 from the above list of support conditions? [Enter 1]

Choice: 1

Please complete the sentence below from the following list of choices:

0 – inconceivable
1 – not likely
2 – possible
3 – probable
4 – almost certain

It is _____ that the E is 1 depending upon . . .

B = 2
D = 2

Choice: 2

INODE C = 3 deleted

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 0

Please pick a component to instantiate:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – Add new component
-1 – Abort

Choice: 0

Please enter the name of the new component:

Item Name: F

F is a new component.

New INODE created: F = No value given!!

REMINDER:

Bayesian Forest Variables are persistent and mutually exclusive. In other words, they take on 1, and only 1, of their possible values. Obviously, this will not accommodate variables that change over time.

Can the value of F vary with time? Y / N: n

Would you like to add another component?: n

Please pick a component to instantiate:

1 – A
2 – B
3 – C
4 – D
5 – E
6 – F
0 – Add new component
-1 – Abort

Choice: 6

Creating new instantiation for F

1 – F = No value given!!

Do you wish to enter a new legal value for F [Enter 0] or choose a pre-existing value [Enter 1]?: 1

Please instantiate F from this menu:

1 – 1
2 – 2
3 – 3
0 – Abort

Choice: 1

New INODE created: F = 1

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 0

Please pick a component to instantiate:

1 – A
2 – B
3 – C
4 – D
5 – E
6 – F
0 – Add new component
-1 – Abort

Choice: 6

Creating new instantiation for F

1 – F = 1

Do you wish to enter a new legal value for F [Enter 0] or choose a pre-existing value [Enter 1]?: 1

Please instantiate F from this menu:

1 – 2
2 – 3
0 – Abort

Choice: 1

New INODE created: F = 2

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 2

Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
6 – F
0 – Abort

Choice: 6

Please pick an instantiation of F:

1 – F = 1
2 – F = 2
0 – Abort
Choice: 1

At present, F's being 1 depends upon the following sets of conditions:

No support conditions!
Enter 0 to add new support conditions for F's being 1 Otherwise, enter 1 to quit: 0

F's being 1
can depend upon which of the following components:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – None of the Above Components
Choice: 1

1 – 1
2 – 2
0 – None of the Above; Abort
Choice: 1

Presently, this condition holds that F's being 1 at time T_i can depend upon the following:

A = 1
Do you wish to extend this condition? Y / N: n

Please complete the sentence below from the following list of choices:

0 – inconceivable
1 – not likely
2 – possible
3 – probable
4 – almost certain
It is _____ that the F is 1 depending upon . . .

A = 1
Choice: 2

This Bayesian Forest is currently inconsistent. Is it correct that F being 2 does not depend on anything else? Y/N: n
Please edit the conditions for F being 2 accordingly.
Bayesian Forest fails consistency checking.

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 2

Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
6 – F
0 – Abort

Choice: 6

Please pick an instantiation of F:

1 – F = 1
2 – F = 2
0 – Abort

Choice: 2

At present, F's being 2 depends upon the following sets of conditions:

No support conditions!

Enter 0 to add new support conditions for F's being 2 Otherwise, enter 1 to quit: 0

F's being 2 can depend upon which of the following components:

1 – A
2 – B
3 – C
4 – D
5 – E
0 – None of the Above Components

Choice: 2

1 – 1
2 – 2
0 – None of the Above; Abort

Choice: 1

Presently, this condition holds that F's being 2 at time T_i can depend upon the following:

B = 1

Do you wish to extend this condition? Y / N: n

Please complete the sentence below from the following list of choices:

0 – inconceivable
1 – not likely
2 – possible
3 – probable
4 – almost certain

It is _____ that the F is 2 depending upon . . .

B = 1

Choice: 3

This Bayesian Forest is inconsistent. Currently, support ranges overlap.
Adjusting ranges for consistency . . .
Conditions were:

It is possible that the F is 1 depending upon . . .

A = 1

It is probable that the F is 2 depending upon . . .

B = 1

New conditions are:

It is not likely that the F is 1 depending upon . . .

A = 1

It is probable that the F is 2 depending upon . . .

B = 1

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 3

Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
6 – F
0 – Abort

Choice: 5

Please pick an instantiation of E:

1 – E = 1
0 – Abort

Choice: 1

At present, E's being 1 depends upon the following sets of conditions:

Support Node #1:

A = 2
D = 1

Support Node #2:

B = 2
D = 2

Please pick one of the preceding sets of support conditions.
Choose 0 to abort

Choice: 1

It is possible that the E is 1 depending upon . . .

A = 2
D = 1

Which part of this support condition do you want to edit?

The term: "possible" [Enter 0]
or
A member of the set of conditions below [Enter 1]?

A = 2
D = 1

Choice: 1

Do you want to add a condition [Choose 0]
or delete one [Choose 1] ?

Choice: 0

Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
6 – F
0 – Abort

Choice: 6

Please pick an instantiation of F:

1 – F = 1
2 – F = 2
0 – Abort

Choice: 2

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 3

Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
6 – F
0 – Abort

Choice: 5

Please pick an instantiation of E:

1 – E = 1
0 – Abort

Choice: 1

At present, E's being 1 depends upon the following sets of conditions:

Support Node #1:

A = 2
D = 1
F = 2

Support Node #2:

B = 2
D = 2

Please pick one of the preceding sets of support conditions.
Choose 0 to abort

Choice: 2

It is possible that the E is 1 depending upon . . .

B = 2
D = 2

Which part of this support condition do you want to edit?

The term: "possible" [Enter 0]
or
A member of the set of conditions below [Enter 1]?

B = 2
D = 2

Choice: 1

Do you want to add a condition [Choose 0]
or delete one [Choose 1] ?

Choice: 0

Please pick a component:

1 – A
2 – B
3 – C
4 – D
5 – E
6 – F
0 – Abort

Choice: 1

Please pick an instantiation of F:

1 – F = 1
2 – F = 2
0 – Abort

Choice: 1

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 5

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 0

Querying BKB...
Correct Answer and solution is Computable.
Validated!

C.3.2 From Figure 4.5.

Please enter the EVIDENCE
Please pick a component:

1 – A
2 – B
3 – C
4 – E
0 – Abort

Choice: 4

Please pick an instantiation of E:

1 – E = 1
2 – E = 2
0 – Abort

Choice: 1

Please enter the EVIDENCE
Please pick a component:

1 – A
2 – B
3 – C
4 – E
0 – Abort

Choice: 0

...calculating connected region...

Please enter the expected ANSWER
Please pick a component:

1 – A
2 – B
3 – C
4 – E
0 – Abort
Choice: 2

Please pick an instantiation of B:

1 – B = 1
2 – B = 2
3 – B = 3
0 – Abort
Choice: 1

Please enter the expected ANSWER
Please pick a component:

1 – A
2 – B
3 – C
4 – E
0 – Abort
Choice: 3

Please pick an instantiation of C:

1 – C = 1
2 – C = 2
0 – Abort
Choice: 1

Please enter the expected ANSWER
Please pick a component:

1 – A
2 – B
3 – C
4 – E
0 – Abort
Choice: 0

Querying BKB...
Incorrect Answer and solution is NOT Computable.

Querying BKB...
Solution known to be correct and it is NOT Computable.

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 1

The assumed components and their assumed states:

E : 1

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 2

Select assumed component and value:

1 – E : 1

Choice: 1

For 'E : 1', one of the following support conditions must be true:

Support condition 1 needs these true:

A : 1
B : 3

However, in the solution, the following were true:

A : 1
B : 1

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 3

Select assumed component and value:

1 – E : 1

Choice: 1

E : 1 is connected to the following components:

A
B

E : 2 is connected to the following components:

A

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 9

****
If any changes are made to the knowledge base*
it must be consistent before continuing. *
****

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation

Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu

Choice: 1

Please pick a component:

1 – A
2 – B
3 – C
4 – E
0 – Abort

Choice: 2

Please pick an instantiation of B:

1 – B = 1
2 – B = 2
3 – B = 3
0 – Abort
Choice: 3
Deleting instantiation of B = 3

It is possible that the E is 1 depending upon . . .

A = 1
B = 3
Do you still wish to delete 3 as valid value for B? Y/N y

Invalidate this entire support condition for E being 1? [Enter 0]
or
Simply remove B being 3 from the above list of support conditions? [Enter 1]

Choice: 1

Please complete the sentence below from the following list of choices:

0 – inconceivable
1 – not likely
2 – possible
3 – probable
4 – almost certain
It is _____ that the E is 1 depending upon . . .

A = 1
Choice: 2

INODE B = 3 deleted

Please select from the following menu:

Instantiations:

0 - Add new instantiation
1 - Delete instantiation
Support Conditions:

2 - Add new support condition
3 - Edit existing support condition
4 - Delete support condition

5 - Return to main menu
Choice: 5

This test case has found an area of incompleteness
in the knowledge base. Choose from the following menu
to aide in locating the problematic knowledge

1 – List Assumptions Made
2 – Inspect an Assumption
3 – List Other Connections in Family

9 – Edit Knowledge Base

0 – Continue Validation

-1 – Skip This Test Case

Choice: 0

Querying BKB...
Correct Answer and solution is Computable.
Validated!

Vita

Howard Gleason was born in October 1968 in Greeley, Colorado. He was awarded a Bachelor of Science in Computer Science from the United States Air Force Academy in 1990. Upon his commission, he was trained as an Imagery Intelligence Officer, and assigned to the (then Foreign Technology Division) National Air Intelligence Center at Wright-Patterson AFB, Ohio. He arrived at the Air Force Institute of Technology in May 1994.

Permanent address: 14274 Skipper Court
Carmel, IN 46033